

University of Warwick institutional repository: <http://go.warwick.ac.uk/wrap>

A Thesis Submitted for the Degree of PhD at the University of Warwick

<http://go.warwick.ac.uk/wrap/3069>

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it. Our policy information is available from the repository home page.

SIMAID: A rapid development methodology for the design of acyclic, bufferless, multi-process and mixed model agile production facilities for spaceframe vehicles.

by

Enrico Tebaldi

**A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy in Manufacturing Engineering.**

**University of Warwick,
Department of Manufacturing Engineering,
October 2001**

Table of Contents

Acknowledgements	1
Declaration	2
Abstract	3
Abbreviations	4
List of figures	5
List of tables	7
 Chapter 1 – Introduction	 9
1.1 Objectives	9
1.1 Background	9
1.2 Agile manufacturing	15
1.3 Designing an agile plant	17
1.4 Problem background	21
1.5 Existing techniques	23
1.5.1 Gradient descent and hill climbing techniques	25
1.5.2 Group Technology	25
1.5.3 Proprietary heuristics	27
1.5.4 Genetic algorithms	28
1.5.5 Tabu search	30
1.5.6 Simulated annealing	31
1.5.7 Networks	34
1.6 Requirements of the necessary methodology	35
1.7 Reasons for selecting the proposed methodology	36
1.7.1 Analysis of best method for layout design	37
1.7.2 Concept justification for SIMAID	38
1.7.2.1 The first stage	39
1.7.2.2 The second stage	40
1.7.2.3 The third stage	41
1.7.2.4 Testing the methodology	42
1.8 Conclusions	43

Chapter 2 – Background	44
2.1 The SALVO programme	44
2.2 SIMAID	46
2.2.1 The SIMAID objective	47
2.2.2 The SIMAID production philosophy	47
2.2.3 Flow control	48
2.2.4 The Tooling	50
2.2.5 Buffers	52
2.2.6 Scheduling	53
2.4 Summary of the main assumptions used in this work	53
 Chapter 3 – Methodology	 55
3.1 The model	55
3.2 the program	57
3.3 Stage I: Initial layout derivation	64
3.3.1 Layout formation	67
3.3.2 The cell	69
3.3.3 The initial layout	70
3.4 Stage II: Scheduling	71
3.4.1 The scheduling problem	71
3.4.2 The SIMAID scheduling solution	71
3.5 Stage III: Layout optimisation by iterative simulation of cell formation	74
3.5.1 The quantity of subassemblies to be simulated	74
3.5.2 Optimisation	75
 Chapter 4 – Principia	 80
Definitions:	80
4.1 Mathematical formulation of Stage I	80
4.1.1 Phase I: Finding the workstation requirements	80
4.1.2 Phase 2: Creating a shortlist by sequence matching	82
4.1.3 Phase 3: Placing the requirements into Groups.	82
4.1.4 Creating loaders/unloaders	84
4.1.5 Cell creation	88
4.1.6 Facilitising with the data	88

4.2 Stage II	91
4.2.1 Introduction	91
4.2.2 The measurement of fitness	93
4.2.3 The prototype SIMAID scheduling GA	94
4.2.4 The gene selection approach	96
4.2.5 Chromosome formation	97
4.2.6 Reproduction	101
4.2.7 The mating mechanism	102
4.2.8 Mutation	102
4.2.9 Iteration number	104
4.3 Stage III.	105
4.3.1 Mathematical formulation	105
4.3.2 Simulation.	107
4.3.3 Modification iterations	109
Chapter 5 – Case studies	117
5.1 Industrial example 1	117
5.1.1 The data	117
5.1.2 The results	118
5.1.3 Critical analysis	119
5.1.4 Assumptions and limitations	120
5.1.5 Discussion	120
5.2 Industrial example 2	122
5.2.1 The data	122
5.2.2 Data translation and incorporation	122
5.2.3 The result	128
5.2.4 Critical analysis and discussion	129
5.3 Industrial example 3	132
5.3.1 The data	132
5.3.2 Data translation and incorporation	132
5.3.3 The results	133
5.3.4 Lamb Technicon's plant layout designs	136
5.3.5 Critical analysis and discussion	137
5.3.6 Conclusion	141

Chapter 6 –Conclusions	142
6.1 Summary	142
6.2 Conclusions	143
6.3 Further work	144
6.3.1 Method	144
6.3.2 Features	144
References	146

Appendix A1	Schematic layout of a current factory, showing the conventional line assembly system.
Appendix A2	Technical drawing of typical spaceframe-based subassembly designed for the SALVO project
Appendix B1	Pseudocode of the real time subassembly routing (dispatching) controller
Appendix B2	Pseudocode of the first cycle of the improvement iterations in stage 3.
Appendix B3	Pseudocode of the second cycle of the improvement iterations in stage 3.
Appendix B4	Pseudocode of the third and final cycle of the improvement iterations in stage 3
Appendix C1	Details of the input data and Lamb Technicon's optimised layout for case study 1.
Appendix C2	Details of the input data and Lamb Technicon's optimised layout for case study 2.
Appendix C3	Details of Lamb Technicon's optimised layout for case study 3.

Acknowledgements

I would like to express my deep gratitude to:

- Dr. Ken Young, my supervisor, for his constant support throughout the project, even when the faint light in the tunnel appeared to be a fading illusion;
- Mr. Kowalewski of Lamb Technicon systems, who provided the data for the industrial cases;
- My girlfriend, friend, confidante and partner Cristina, for the best moral and material support a guy could ever wish for;
- My parents, brother and sister, who have patiently waited this long even though I kept predicting “just 3 more months ...”;
- Many other friends and colleagues who have helped in one way or another.

Declaration

I declare that the work presented herein is my own work and has not been submitted to another institution for award.

Enrico Tebaldi

Abstract

The facility layout problem (FLP) is a non-linear, NP-complete problem whose complexity is derived from the vast solution space generated by multiple variables and interdependent factors. For reconfigurable, agile facilities the problem is compounded by parallelism (simultaneity of operations) and scheduling issues. Previous work has either concentrated on conventional (linear or branched) facility layout design, or has not considered the issues of agile, reconfigurable facilities and scheduling. This work is the first comprehensive methodology incorporating the design and scheduling of parallel cellular facilities for the purpose of easy and rapid reconfiguration in the increasingly demanding world of agile manufacturing. A novel three-stage algorithm is described for the design of acyclic (asynchronous), bufferless, parallel, multi-process and mixed-model production facilities for spaceframe-based vehicles. Data input begins with vehicle part processing and volume requirements from multiple models and includes time, budget and space constraints. The algorithm consists of a powerful combination of a guided cell formation stage, iterative solution improvement searches and design stage scheduling. The improvement iterations utilise a modified (rules-based) Tabu search applied to a constant-flow group technology, while the design stage scheduling is done by the use of genetic algorithms. The objective-based solution optimisation direction is not random but guided, based on measurement criteria from simulation. The end product is the selection and graphic presentation of the best solution out of a database of feasible ones. The case is presented in the form of an executable program and three real world industrial examples are included. The results provide evidence that good solutions can be found to this new type and size of heavily constrained problem within a reasonable amount of time.

Abbreviations

- FLP Facility layout problem;
- QAP Quadratic assignment problem;
- MHS Material handling system;
- FMS Flexible manufacturing system;
- CPU Central Processing Unit;
- GT Group technology;
- MPCF Machine-part cell formation;
- CASE Clustering algorithm for sequence data;
- TAL Threshold affinity level;
- GA Genetic algorithms;
- SPP Shortest path problem;
- MFP Maximum flow problem;
- MSTP Minimum spanning tree problem;
- NN Neural networks;
- AGV Automatic Guided Vehicles;
- SPR Self-piece riveting;
- AI Artificial intelligence;
- ES Expert systems;
- SPM Single point mutation
- GUI Graphical user interface

List of figures

Figure 1.1	The state of current vehicle manufacturing.	11
Figure 1.2	Vehicle production capacity and maximum annual production	13
Figure 1.3	The three components of service time.	14
Figure 1.4a	A diagrammatic 'cell matrix' approach to production.	17
Figure 1.4b	A schematic 'cell matrix' approach to production.	18
Figure 1.5	The probability shapes of several search methods	37
Figure 2.1	A possible example of a space frame vehicle body structure.	45
Figure 2.2	The 5 main types of part/subassembly flow.	48
Figure 2.3	The SALVO prototype mobile tooling.	50
Figure 2.4	The SALVO prototype production facility.	51
Figure 2.5	A hypothetical agile facility.	52
Figure 3.1	The overall SIMAID model.	55
Figure 3.2	Top-level flowchart summarising SIMAID's functionality.	56
Figure 3.3	The main SIMAID window	57
Figure 3.4	The SIMAID robots window	58
Figure 3.5	The SIMAID robot selection criteria window.	58
Figure 3.6	The SIMAID process data window	59
Figure 3.7	The SIMAID process data follow-up window.	60
Figure 3.8	Volume data window.	61
Figure 3.9	The costs database window.	62
Figure 3.10	The constraints data input window.	63
Figure 3.11	The cell data window.	64
Figure 3.12A	Flowchart of the first stage of the methodology (part A).	65
Figure 3.12B	Flowchart of the first stage of the methodology (part B).	66
Figure 3.13	The process-placing algorithm in phase three of stage 1.	67
Figure 3.14	Example of the sequence-adding algorithm.	68
Figure 3.15	Incremental adding of processes with different requirements.	68
Figure 3.16	The layout of a hypothetical cell.	69
Figure 3.17	Diagram showing the 'twice-the-cell-number' concept.	73
Figure 3.18	Flowchart showing a general overview of Stage II.	73
Figure 3.19	Flowchart showing a general overview of the warm-up period.	75
Figure 3.20	The final stage leading to the facility layout generation.	77

Figure 4.1	Example of placement of processes into groups.	83
Figure 4.2	Example of potential process duplication in the groups.	83
Figure 4.3	Example of the movement of processes between groups.	84
Figure 4.4	Example of a design of the initial facility.	90
Figure 4.5	A hypothetical facility design.	91
Figure 4.6	Flowchart of the second stage of the SIMAID methodology.	95
Figure 4.7	Subassembly routing despatch control.	108
Figure 4.8	The first cycle of improvements iterations.	110
Figure 4.9	A hypothetical initial facility layout before and after changes.	111
Figure 4.10	The facility design following the addition of a further process.	111
Figure 4.11	The second cycle of improvements iterations.	113
Figure 4.12	The third and final cycle of improvements iterations.	114
Figure 4.13	First example of possible third cycle changes.	115
Figure 4.14	Second example of possible third cycle changes.	115
Figure 4.15	Third example of possible third cycle changes.	115
Figure 5.1	The facility layout generated by SIMAID.	118
Figure 5.2	The facility layout generated at 240,000 units per annum.	118
Figure 5.3	The actual SIMAID data window.	124
Figure 5.4	The process selection window.	125
Figure 5.5	Cost data used for the exercise.	126
Figure 5.6	The constraints window.	127
Figure 5.7	The cell data window.	128
Figure 5.8	The results window.	129
Figure 5.9	The overall process sequence.	133
Figure 5.10	The end result of the third case.	134
Figure 5.11	The solution time distribution	135
Figure 5.12	Schematic diagram of the original Lamb design for the plant.	137

List of tables

Table 1.1	The typical part-process matrix.	21
Table 1.2	The 10 main criteria necessary for speedy facility layout design	36
Table 4.1	Example of a subassembly requiring the processing of 5 different joints.	81
Table 4.2	A hypothetical production set.	95
Table 5.1	Subassembly data inputted into SIMAID.	117
Table 5.2	The cells' utilisation rates.	119
Table 5.3	Original process data.	123
Table 5.4	Translated process data (case study 2).	123
Table 5.5	Translated process data (case study 3).	124
Table 5.6	The cell utilisation levels.	128
Table 5.7	The processes required.	133
Table 5.8	Original process data for the front structure assembly zone.	133
Table 5.9	The results from the simulation.	135
Table 5.10	Original process data for the front structure assembly zone.	136
Table 5.11	Original process data for the panel dash assembly zone.	136

Chapter 1 - Introduction

1.1 Objectives

This work describes a methodology for the rapid design and modification of future agile, space frame-based production facilities, through a combination of established, modified and novel solution search techniques and a highly visual step-by-step GUI. The aim is to enable production planners or facilities engineers to design new production layouts from a set of product processing data, financial objectives and layout constraints. Its main advantage over the standard design procedures¹ currently used in industry, is its rapid execution, typically less than 30 minutes from data input to layout output.

The program is designed to transform a set of vehicle production data to produce a viable layout design. The layout will be shown as a 2-D plan of a group of production cells, where process operations on the vehicle subassemblies (such as spot welding or riveting) take place. The data set used comprises the vehicle's processing requirements, volume, sequence constraints and the area required. Multiple vehicle assembly is a complex process and subassembly scheduling is important to the final layout's efficiency. This methodology provides for simultaneous layout design and scheduling. The advantage of this method rests in the evolution of the layout being influenced by both activities, substantially improving the result.

1.2 Background.

Through rising product development costs, sharp, focussed competition and an increasingly discerning public, mass-market (not niche) car manufacturers are less likely to be able to remain financially profitable just by producing attractive products. The current wave of consolidation seen in the automobile industry quite clearly points to the way forward: globalisation. *"I think we will get to the point where even someone selling a million vehicles a year will be a niche player"*, said Jacques Nasser of Ford [ANE, 1998]. No company that builds less than one million units a year will be able to survive for long in the mainstream volume segment [ANE, 15 Feb 1999].

The niche market is important – 20% of all vehicles sold in Europe in 1998 were not from mainstream categories [ANE, 1 Feb 1999] – but such specialisation may lead to narrow

¹ Current industry-wide design steps: manual workstation number calculations, CAD layout design, then exporting it to cell-level and/or facility level simulation packages such as Robcad and Witness, respectively.

segment dependency and higher exposure to downturns and fickle market forces. The recent flurry of mergers and acquisitions leaves no doubt of the mainstream manufacturers' drive to attenuate this effect by ensuring their presence in as many market segments as possible.

There are seven 'large' manufacturers present today (together with five 'middle-sized' makers), and it is expected that this number will reduce substantially within ten years. The scenario being unveiled is one of a much smaller number of 'private transportation' conglomerates, producing a wide range of vehicles on every continent of the world. It makes sense to build locally, for both logistic and marketing reasons; hence it is likely that these groups will build several models as 'world cars', with variation depending on local preferences.

The implications are that the larger conglomerates (unlike niche players), without serious brand differentiation or any great technological leads over competitors, will need to produce:

- Many more models,
- In increasingly shorter lead times, and
- With greater and greater variety in terms of styles and content;

With current methods, the development costs incurred in running such programmes are bound to increase. Increasing demand for new models, and the corresponding decrease in individual model lifetime means that they effectively need to concurrently run the equivalent of dozens of individual companies (divisions) for their chosen niche segments. One of the limitations currently found in manufacturing is the way vehicles are produced. Current vehicles are being assembled in ubiquitous 'lines', where parts are added as necessary, starting from the basic body structure, then proceeding through a paint oven, and finally receiving content such as seating, glazing, dashboard, etc., as shown in the schematic diagram below (Fig. 1.1). To avoid additional costs, (partly due to research and development duplication between divisions and geographical zones), it would make sense to 'share' some of the design and production. For production, this would mean sharing the same plant between several models.

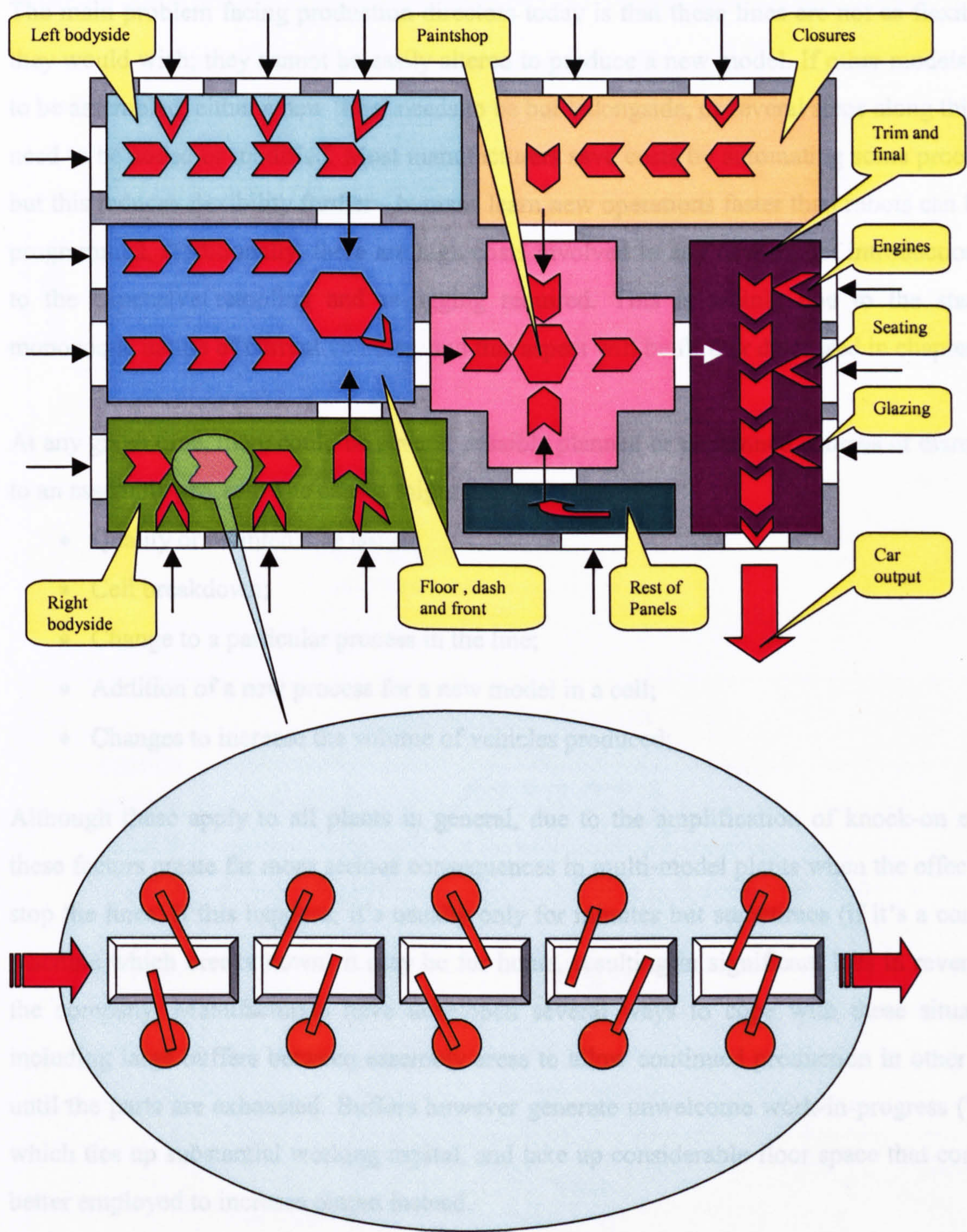


Fig 1.1 The current state of vehicle manufacturing. Top: A schematic of a generalised assembly plant, showing the sequence of assembly processes leading to complete car assembly. The black arrows indicate the entry point for parts (or pre-built subassemblies), and the red arrows indicate the internal assembly processes. Above: A schematic diagram showing the typical assembly line, with robots on either side of a central moving assembly area, usually on a conveyor (the robots may have identical or different processes). The flow is unidirectional and each operation needs to be completed within the facility-wide cycle time. Appendix A1 shows the layout of a current plant, with the conventional line assembly system.

The main problem facing production directors today is that these lines are not as flexible as they would wish; they cannot be easily altered to produce a new model. If other models need to be assembled, either a new 'line' needs to be built alongside, or several steps along this line need to be added or modified. Most manufacturers save costs by automating some processes, but this reduces flexibility further - humans learn new operations faster than robots can be re-programmed. Additionally, there are high costs involved in any new model introduction due to the expensive retooling and re-jigging required. This is mainly due to the standard monocoque nature of current vehicles, and this aspect will be further discussed in chapter 2.

At any given time, there could be several possible planned or unplanned sources of disruption to an assembly process. The causes might be:

- Quality or maintenance issues;
- Cell breakdown;
- Change to a particular process in the line;
- Addition of a new process for a new model in a cell;
- Changes to increase the volume of vehicles produced;

Although these apply to all plants in general, due to the amplification of knock-on effects these factors create far more serious consequences in multi-model plants when the effect is to stop the lines. If this happens, it's usually only for minutes but sometimes (if it's a complex machine which breaks down) it may be for hours, resulting in significant loss in revenue to the company. Manufacturers have developed several ways to cope with these situations, including large buffers between assembly areas to allow continued production in other areas until the parts are exhausted. Buffers however generate unwelcome work-in-progress (WIP), which ties up substantial working capital, and take up considerable floor space that could be better employed to increase output instead.

Plants have been designed to perform reasonably efficiently, with uptime usually targeted at around the 85% mark (Holweg and Jones, 2001). To address the unwelcome downtime, equipment vendors, company repairmen and the necessary replacement equipment have been put on continuous standby for such events. Historically, manufacturers have always over-

facilitised² (built excess capacity into) their plants, as can be seen in the Rover and Ford examples below [Society of Motor Manufacturers and Traders, Annual report 2001]:

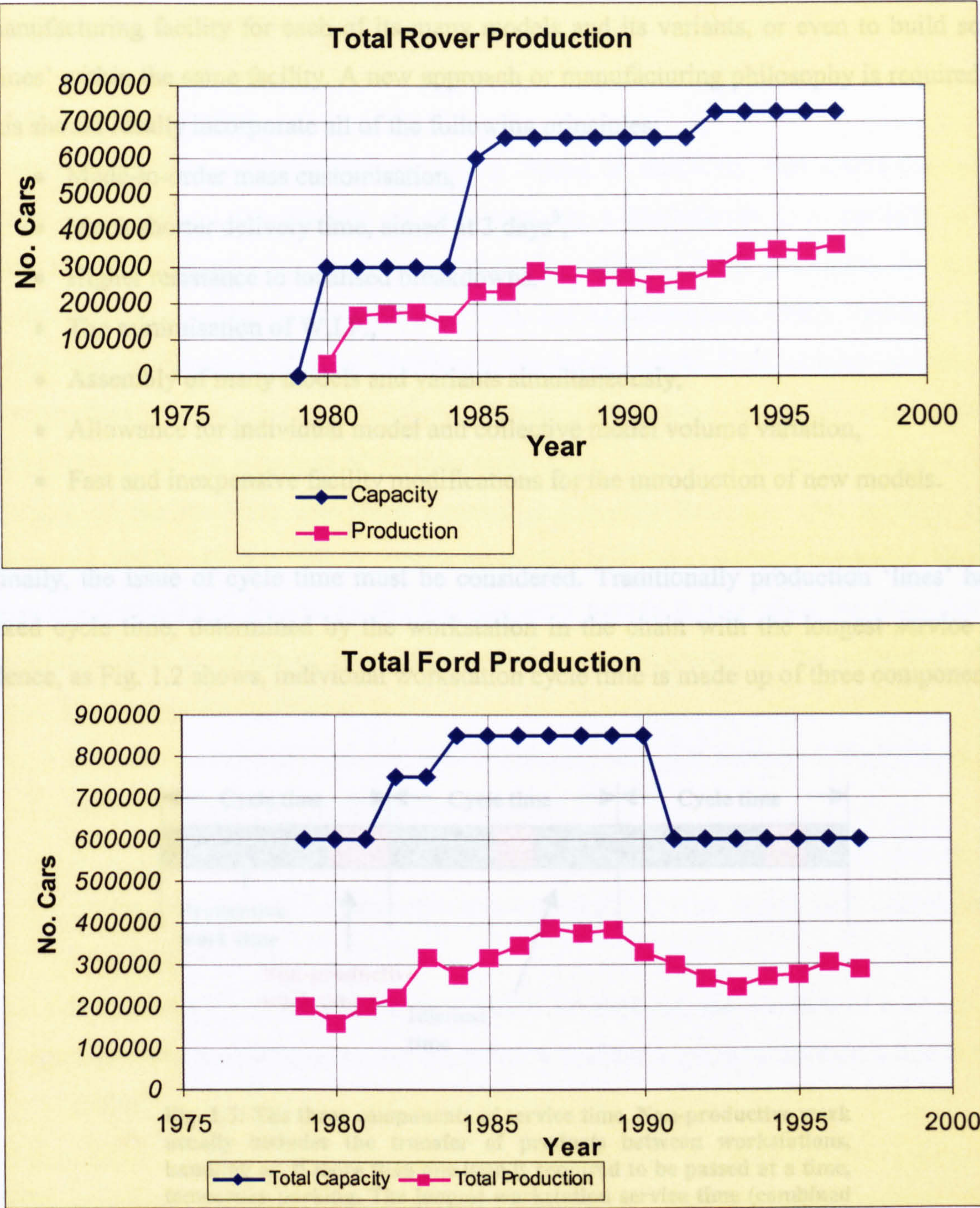


Fig. 1.2: Estimated total vehicle production capacity and maximum annual production for Ford and Rover (Society of Motor Manufacturers and Traders, Annual report 2001).

On higher demand, production levels are raised, although at substantial extra cost. In times of low demand, the unused excess capacity constitutes unutilised capital, which could be

² ‘Facilitisation’ is the industry’s term for the incorporation in a production plant of equipment (jigs and fixtures, robots, material handling systems, etc) necessary for assembling the required volume of vehicles.

better spent on other money-generating resources. None of these solutions, are ideal, as stopping a production facility is not in the interests of the company. Additionally in the above-described scenario it would be cost-prohibitive for any company to build a different manufacturing facility for each of its many models and its variants, or even to build several 'lines' within the same facility. A new approach or manufacturing philosophy is required, and this should ideally incorporate all of the following principles:

- Made-to-order mass customisation,
- Much shorter delivery time, aimed at 3 days³,
- Higher resistance to localised breakdowns,
- The minimisation of W.I.P.,
- Assembly of many models and variants simultaneously,
- Allowance for individual model and collective model volume variation,
- Fast and inexpensive facility modifications for the introduction of new models.

Finally, the issue of cycle time must be considered. Traditionally production 'lines' have a fixed cycle time, determined by the workstation in the chain with the longest service time. Hence, as Fig. 1.2 shows, individual workstation cycle time is made up of three components:

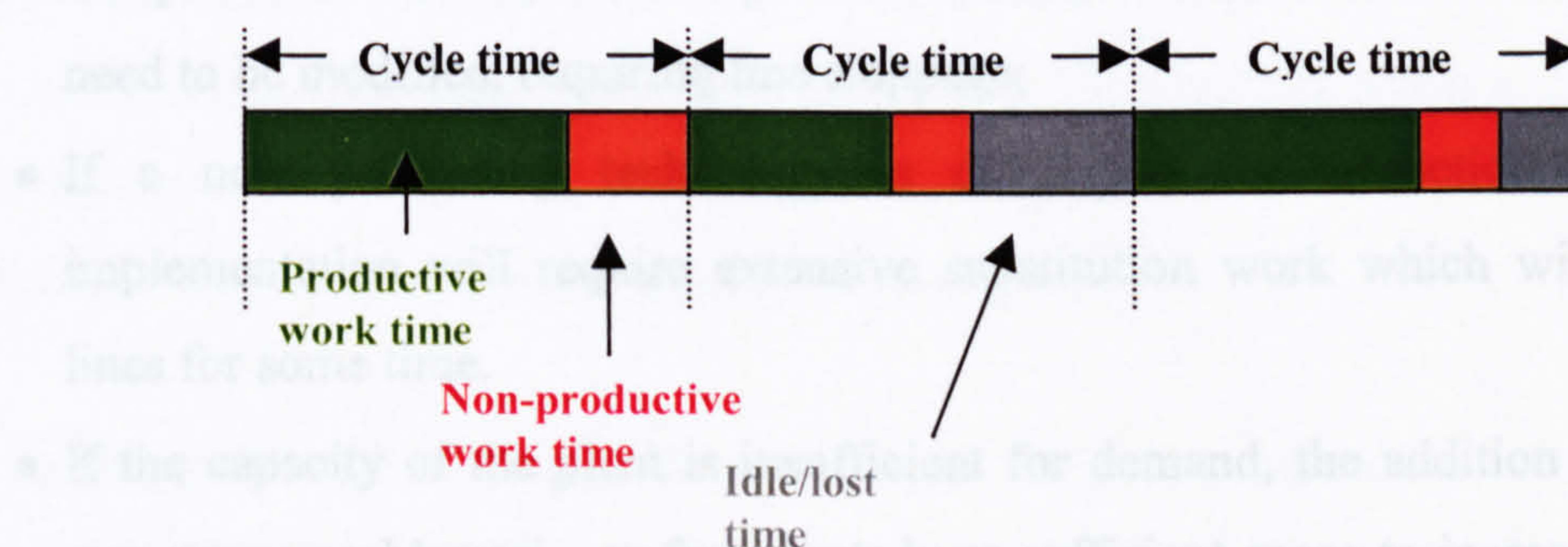


Fig. 1.3: The three components of service time. Non-productive work usually includes the transfer of products between workstations, handling or, if more than one item is required to be passed at a time, temporary parking. The longest workstation service time (combined productive and non-productive times) becomes the whole line's cycle time (Wild, 1995).

When many such workstations are placed together, it is inevitable that, even after line balancing there will be slack (idle) time. If several models are to be assembled on the same line, the problem would be accentuated further: Process incompatibility would result in

³ From the ICDP report, 'Fulfilling the promise: What future for franchised car distribution?' by the 3DayCar programme consortium. WWW.cardiff.ac.uk/3daycar.

increased delays due to workstations being blocked or starved while waiting for previous subassemblies to finish processing. Should at any point any of the workstations need changing, a substantial balancing loss would be incurred for all models, requiring re-scheduling and perhaps even line process re-sequencing.

1.2 Agile manufacturing

Made-to-order mass customisation requires a degree of flexibility well above current best practice. At present, the majority of the major car manufacturers have partially or fully succeeded in implementing *flexible* manufacturing system (FMS) principles, for example Volkswagen (Anon, The Industrial Robot, 1989), Toyota (Masuyama, 1995), Vauxhall (Farish, 1995). The Volkswagen plant at Wolfsburg can build several Golf variants and the more modern plant at Emden can assemble Golfs, Passats and Jettas (a saloon Golf design). FMS are usually based on the principle of flexible automation of tasks, such as assembly; a typical example of the available automated systems is Comau's Robogate spot welding system (Mattucci, 1994), which is able to operate on several body types without re-jigging. However, even this flexibility is limited by the fact that even though it may allow for more than one component, part or subassembly to be processed, it is still limited by the designer's original input. The reasons are as follows:

- If a part is modified (or a new part introduced) then that section of the FMS may also need to be modified, requiring line stoppage;
- If a new processing technology is developed for a section of the body, its implementation will require extensive substitution work which will immobilise the lines for some time.
- If the capacity of the plant is insufficient for demand, the addition of new processes may prove problematic, as few plants have sufficient space to increase a line in length or size.

FMS practices are currently insufficient on their own for companies to compete successfully in a rapidly changing environment. In order to address these shortcomings, a plant must be construed from principles that allow changes to be implemented rapidly and without affecting current production, whether new component introduction or process addition – effectively building in 'agility', or responsiveness. Agility has been defined in many ways. In terms of a facility its essence can be summed up in four key concepts (Kidd, 1996):

- Adaptability – Possessing the capability to reconfigure itself with ease;

- Speed – Implementation of any required changes within an acceptable frame of time;
- Robust – Implementation of any required changes without undue (and costly) disturbances to its surroundings;
- Dynamic – Actively managing the changes ahead of requirement and in order to forestall future problems.

There is a clear need for companies to respond rapidly to product change and time-to-market. Re-thinking the manufacturing process by designing an agile production facility is part of this picture. The facility should be able to deliver specific customer orders (mass customisation), selected from a variety of models and variants to the customer within days. Current delivery targets are 10 to 15 days for the bigger manufacturers, aiming to cut the present average delivery time of 48 days (Holweg and Jones, 2001). Ideally, a customer should be able to receive his vehicle within a nominal but achievable 3 days, from the customer entering the showroom to being able to insert his key into the ignition.

The above picture implies a range of implications for the manufacturer and supply chain, from the use of concurrent engineering in every aspect of the business, to the application of specific technologies and practices such as design for assembly, sub-system modularity, component/platform sharing, use of late configuration principles, as well as competent design and management of an agile manufacturing facility. Finally, it is worth adding that without a responsive process of logistics, enabling the right type and number of components to arrive to trackside as and when necessary, agility is meaningless.

The agile aspect will be able to do this despite variable economic conditions, a constantly changing marketplace and frequent model changes. The long-term vision is the construction of a plant in which 10 models are simultaneously produced, and where the specific model mix is unlikely to be constant from month to month. The facility will thus need to be sufficiently adaptable to enable the seamless implementation of model facelifts, variants and new model introductions.

1.3 Designing an agile plant.

The influence of facility design on overall costs cannot be underestimated. Several authors have carried out performance-cost analyses (Tompkins and White, 1984, and Sule, 1988) and pointed out that 20% - 50% of the total operating expenses in manufacturing are attributed to layout-related costs, concluding that effective layout design can save at least 30% in ongoing costs (Chiang and Kouvelis, 1996). The introduction of agile plants, with constantly changing layouts and cell contents (processes), should increase these savings.

The conventional Body-In-White (BIW) line assembly system, applied in one form or another since the 1920's, appears to be reaching the limits of its capability in terms of flexibility. From experience, manufacturers have regularly found it too expensive for frequent changes. An alternative could be parallel production facilities, where several lines coexist in parallel each producing a set of car models, or families. However, such facilities involve an unnecessary level of expensive duplication and do not give benefits for unpredictable market demand. If demand for one type of vehicle decreases or increases, one or more lines will be either under-utilised or at capacity; i.e. unable to fulfil demand. A better solution is a production facility where a *matrix* of *cells* are able to accept every model and its variants, with each cell able to process a set of subassemblies belonging to the models. The idea is described in fig. 1.4a and 1.4b, below:

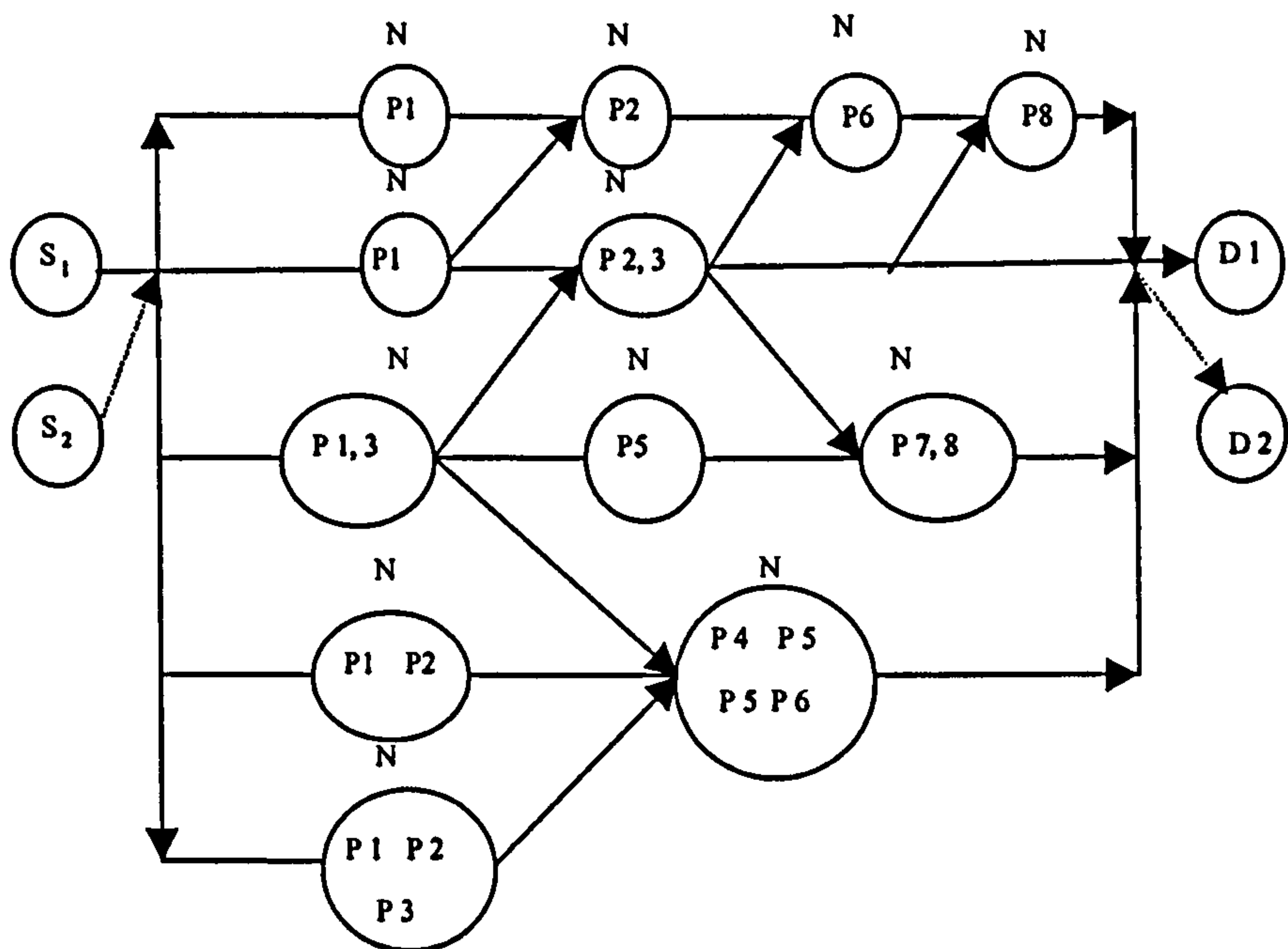


Fig 1.4a The 'cell matrix' approach to subassembly production. The processes within cells are shown as P1, P2 ... and the sources (parts loaders) as S1. D1 is the destination (such as a paint oven or framing station). There may be more than one source or destination, depending on the number of models being assembled.

This cell matrix needs to be designed so as to include all of the models' process requirements, as well as their volumes, which, for mass producers today, can be as much as one million vehicles a year per plant. In schematic form it can be represented like this:

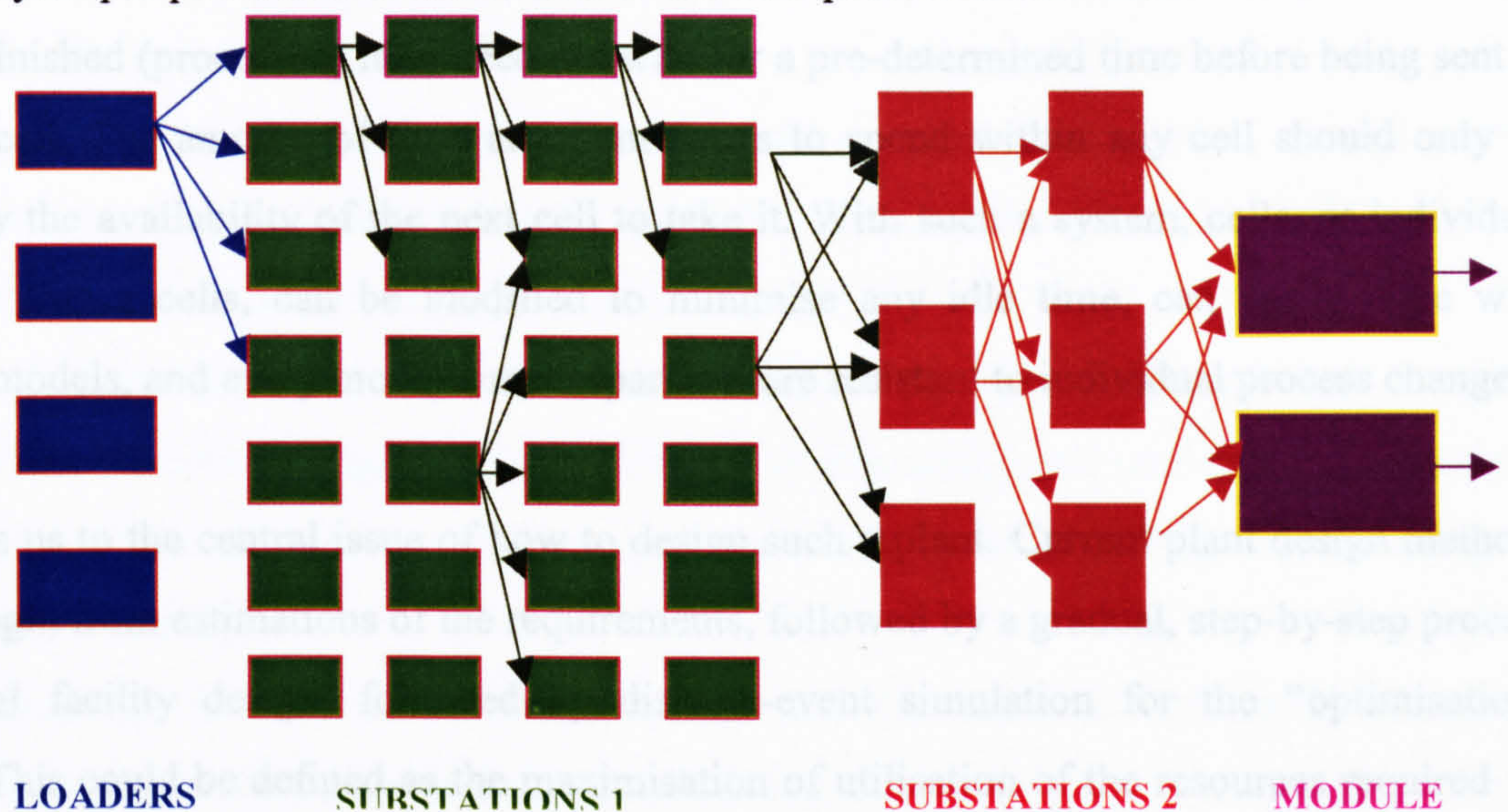


Fig 1.4b The 'cell matrix' approach, schematic representation.

In this work, a *cell* is defined as a unit of space in which a set of processes $\{p_1, p_2, p_3 \dots p_n\}$, for example welding or riveting, have been placed in close proximity in order to minimise the distance travelled and time taken for the set of subassemblies going through it. Fig 3.16 shows a possible cell as applied to a spaceframe assembly. Ideally, the subassemblies should not be required to be moved within such a cell, but would instead be held by a fixture or tooling in such a way as to enable any of the cell's processes to function correctly as required. The material handling system (MHS), which delivers the subassemblies and parts to the cells, should also be designed with this in mind.

Considering the above, there is a need to minimise or maximise several objectives, some of them conflicting. The usual objectives are to minimise total facility costs, makespan, intercellular moves, etc. and maximise cell utilisation rates, efficiency, and flexibility. Additional 'desirables' would be having a degree of cell breakdown resistance, volume (capacity) flexibility, etc. There may be other factors to consider depending on what is required. For instance, when building a new green-field plant, considering available space is not usually a top priority. However if using part of an existing factory, space is at a premium, and the 'space' objective becomes more important.

On the issue of cycle time, traditional lines operate at a time loss due to idle time. In order to minimise this situation, it would be preferable if the matrix cell system did not operate on such a cycle time. Different cells will have different individual service times, but in the cell matrix case the finished (processed) item need not wait for a pre-determined time before being sent to the next cell. The amount of time any item needs to spend within any cell should only be limited by the availability of the next cell to take it. With such a system, cells, or individual processes within cells, can be modified to minimise any idle time, can easily cope with different models, and every model's makespan is more resistant to individual process changes.

This leads us to the central issue of how to design such a plant. Current plant design methods tend to begin from estimations of the requirements, followed by a gradual, step-by-step process of manual facility design, followed by discrete-event simulation for the "optimisation" process. This could be defined as the maximisation of utilisation of the resources required for the models' assembly, i.e. the allocation of the correct amount of equipment to do the job, at the minimum cost. The whole process usually takes a few weeks, even for simple sub-assembly facility design, and for one or more complete models, can take several months. The problem also applies to any reconfiguration that may need to be done. With time-to-market becoming an increasingly important weapon in the 'car wars', this is no longer sufficient and a new, far quicker method is required, in the form of a new software 'tool'. Ideally, this tool should incorporate all of these features:

- Ability to design an optimal, or good sub-optimal agile facility for as many models as is required, from simple part and process data;
- Ability to shape the design according to certain user-input localised criteria and budget constraints;
- Ability to give adequate visual feedback in an easy to understand manner;
- Allow the user to go back at any time and modify these criteria when circumstances change, such as the introduction of a new model,
- Ability to deliver the above objectives in minutes.

Finally, an important fact to consider is that in terms of a flexible facility design, the concept of optimality cannot have much relevance. Fluctuating market forces, usually out of a manufacturer's control, prone to temporary fads and trends, may severely affect any individual model or model class desirability, and thus create an unpredictable demand in terms of both total vehicles and individual model sold. Part of the answer to this is mass-customisation,

which is rapidly becoming an essential competitive weapon. A multi-model facility design, which could perhaps be defined as optimal for a given set of models and their respective volumes *today*, would inevitably be sub-optimal when this production configuration changes *next month*. The three main reasons are:

- Increasing frequency of current model facelifts and the introduction of new models means cell processes, whole cells and vehicle routing and scheduling will need to be changed at irregular but frequent intervals;
- Within the same model, mass customisation will mean individual unit assembly variation will be huge, inevitably leading to fluctuating cell (and process) utilisation rates; this in turn may drive process addition or removal, repositioning or modification;
- Increasing rate of change in technology and the quest for efficiency will mean processes will become obsolete faster, leading to more frequent replacement.

Solution to this could be either a continuously self-configuring facility, which is rearranged every time a factor changes, or a sub-optimal facility whose characteristics are sufficiently broad to allow cost-effective production over a whole range of factor variations. Both solutions have their problems. The first one, apart from the engineering complexities involved, is seen to be too expensive for most car manufacturers to contemplate. The second solution is more cost effective and accessible, but clearly not ideal. This leads to a slightly different formulation of the problem to be defined. The objective now becomes:

"To design an agile facility which incorporates an optimality over time as close as possible to a true (and changing) optimal value, without excessive cost and complexity".

This manufacturing agility, defined here as reconfigurable flexibility, implies the use of a cellular approach, as independent cells have the advantage of being accessible for changes without unnecessary disruption to the rest of the facility. It also follows that in order for parts to be assembled into vehicles, a comprehensive MHS must be in place linking these cells as well as parts source(s) and sub-assembly sink(s). These correspond, in the car manufacturing industry, to a parts loader and a framing station, where the various sub-assemblies of a vehicle are put together under controlled conditions to make the basic body framework of a car.

1.4 Problem background

There are s sub-assemblies per model m , with n parts in a subassembly with j joints, each of which may require to be processed in p different ways. Thus, potentially there may be:

$$\sum_m^M s \cdot n \cdot j \cdot p$$

possibilities in terms of processing combinations. As a simple example, if we wish to make 4 models in one facility, each model having 4 subassemblies, each subassembly 4 parts, each part 4 joints, each of which may need up to 4 different processes, then there would be 1024 combinations. In reality in model families there is considerable sharing of cross-model parts and a high degree of similar processing requirements, hence the actual number of combinations tends to be far smaller. However, as conventional vehicles today may have tens of subassemblies, each with thousands of parts, it can be seen that build complexity can nevertheless easily reach unmanageable proportions within a single production line, giving the need for separate lines.

The use of cellular facilities, while potentially alleviating this problem, brings challenges of its own. The design of multi-model cellular facilities must cope with individual model volumes, part/process commonality, sequences, cell distances and several other factors, which affect the final design and subsequent running costs. Some of these are solved through process 'pooling' for common part families, as described further under 'Group technology'. Even so, some models will have at least one unique processing requirement. Often but not always, a specific processing order is necessary between parts; hence creating a 'part-process' matrix is a useful way of summarising the operations required in the facility. Table 1.1 below shows an example

	Part A	Part B	Part C	Part D	Part E
Process 1	1	0	1	1	0
Process 2	0	0	0	1	1
Process 3	1	1	1	0	0
Process 4	0	1	0	1	0
Process 5	1	1	1	0	0
Process 6	1	0	0	0	1

Table 1.1 The typical part-process matrix. In this case part A will require processes 1,3,5 and 6; these processing requirements are similar to part C's, so a cell may consist of these relevant processes put together and be visited by those parts. Note that part E has a unique processing requirement, as no other part requires the use of process 6.

Most approaches to date have used this matrix to formulate viable initial cell configurations, subsequently improving them by minimising either cell formation costs, or intercell part travel distances (Kusiak, and Cheng, 1991). This is often done by the using Group Technology technique, as described below.

While such approaches are useful, they do not address the problem completely. As seen above, there are other factors that a facility could be optimised by, or at least considered and perhaps included in the optimisation process. Although full optimisation remains both technically elusive and operationally irrelevant to future agile facilities, the aim must still be that of designing a layout as close as possible to the nominally 'optimal' one for any given scenario.

Given the complexity of this task, current methodologies tend to perform inadequately. Matrix-based solutions such as that described above ignore operational issues including cell state, performance or structures. These in real life are important factors in design (Salum, 2000). Most authors' attempts at minimising intercell travel distances (important both in terms of material handling systems (MHS) costs and space required) do not consider intercell travel times. Many approaches place a limit of one operation for each part type, or assign each operation to one machine only. Although many approaches use capacity constraints, they do not consider workload distribution, despite the well-described effect this has on facility production efficiency (Lozano et al, 1999).

Scheduling, while extensively researched on its own, is routinely ignored in facility layout determination problems, and determination of specific cell layout in the available space is also rare. Cell utilisation is frequently taken as a constraint but usually not in conjunction with two other important factors in the car manufacturing industry: Vehicle volume flexibility and cell (or machine) breakdown resistance. To date, a major drawback of cell formation techniques has been their inability to lay machines in cells on a shop floor, leaving 'cell formation' as abstract concept (Salum, 2000). All of these factors must be balanced against an overall facility design cost, which is given to planners in industry in the form of a total construction budget.

There is a clear requirement for an algorithm, or a collection of algorithms, which addresses these issues. Due to the complexities involved, this cannot be a purely mathematical approach. Heuristics may be used, but the cell layout (design) guidance mechanism must incorporate all

the factors relevant to a facility's design, running, maintenance and subsequent reconfigurability. This then becomes a multi-criteria optimisation problem, where many of the objectives may be conflicting. Below are a selection of existing techniques and some examples of previous work in these fields. Although by no means exhaustive, the following section is representative of the strengths and limitations of existing procedures and the necessity for a different, more holistic approach to the issue of production facility design.

1.5 Existing techniques

There has been a great amount of work done in this area, using a variety of techniques. The facility layout problem (FLP) is essentially a quadratic assignment problem (QAP) where the aim is to find the optimal distribution of m machines in c cells such that a quantity n of parts can be adequately processed. The objectives usually consist of minimising the cost C of such a layout and/or minimising makespan:

$$\text{Minimize } C = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sum_{l=1}^n a_{ijkl} \cdot X_{ij} \cdot X_{kl}$$

$$\text{Subject to: } \sum_{i=1}^n X_{ij} = 1 \quad j = 1, \dots, n$$

$$\sum_{j=1}^n X_{ij} = 1 \quad i = 1, \dots, n$$

Where: $X_{ij} = 1$ if facility i is assigned to location j , 0 otherwise;
 a_{ijkl} = the cost of locating facility i at location j and facility k at location l .

Several other objectives can also be applied, such as maximising cell utilisation, minimising material handling or average tardiness (Chen and Sha, 1999). When applied to a multi-model multi-process situation, such a multi-objective facility layout optimisation problem has been shown to be NP-hard/NP-complete. This is short for Non-deterministic Polynomial time-hard/complete (Kusiak and Heragu, 1987), and if approached mathematically, will lead to an intractable problem. NP-complete problems are those that can be solved in polynomial time, using the 'brute force' approach, or checking every possible available combination. In practice, this is often not feasible, as the number of combinations can be very large, making searching for an exact solution an exponential problem. A recognised limit for the possibility of finding

optimality by such a process is 15 machines, above which only approximations to optimality can be guaranteed (Chiang and Kouvelis, 1996).

A computationally efficient, general algorithm to optimally solve the facility layout problem regardless of size has not been found to date. Facility design is a non-linear, multi-objective and multi-parameter problem and it does not lend itself well to current techniques. Past efforts have considered integer programming for cell formation and machine selection, together with cell routing for parts (Rajamani *et al*, 1990). Such attempts reach optimal solutions for small, simple problems but for larger problem sizes (as found in industry) the computer requirements, in terms of CPU power and solution time, become prohibitive. This has led to the conception of 'heuristics'; rule-based algorithms in which optimality is not guaranteed, but the good sub-optimal solutions generated are seen as acceptable for most situations in the real world.

To give an indication of the scale of the problem, assume that if the probability of finding the global minimum is 1, then it will be found, and if it is 0, then it cannot be found. If there are N possible solutions, one of which is the optimal one, and only half are examined, the probability of finding the global minimum would be 0.5. If the number of solutions searched is n , the probability of finding the global minimum would be

$$P(g) = n / N$$

The probability of finding the global minimum in n searches is:

$$P(g) = 1 - ((N-1) / N)^n$$

Thus, the probability of finding the global minimum in a solution space of 10,000 points in 100 iterations is:

$$P(g) = 1 - ((10,000 - 1) / 10,000)^{100}$$

$$P(g) = 0.0099$$

Such methods work well when the problem size is small but are difficult to implement in problem categories such as facility design, where the problem space can be huge.

1.5.1 Gradient descent and hill climbing techniques

Generally applied to linear problems, these techniques use the concept of probabilistic search to arrive at solutions. Initially it randomly selects a point in the solution space and reads its value. However, its subsequent search technique isn't random: It assesses its next point and if this has a value closer to the desired optimum (whether higher or lower), this value is kept and the previous one discarded. As long as with each subsequent measurement a lower value is found, the quality of the solution keeps improving and the direction keeps moving towards the global maximum (minimum). Gradient descent is a mathematically proven technique that will always find the optimum as long as all possible alternatives are searched and evaluated and the problem formulation is linear.

Hill climbing techniques vary from gradient descent methods slightly. Designed for quadratics, they also select random points but then look at the values of neighbouring points as well. If the neighbour's value is less than the current value, then that value becomes the next target value to beat. This technique tends to be much faster than gradient descent or random search for a few iterations, and has a good chance of finding the optimum as long as the search intervals are chosen judiciously. However, it also suffers from the trap of falling into local minima about fifty percent of the time. A variation of hill climbing is *steepest descent*, where all the local neighbours are examined and the one with the lowest relative value is chosen first. This technique tends to be even faster than hill climbing but also suffers from the local minima trap. Generally, hill climbing and the similar techniques are limited to solving quadratics, and more recent, intelligent algorithms (such as SA and GA) have been devised which are not limited by assumptions about linear factors.

1.5.2 Group Technology

Group technology (GT) is a manufacturing philosophy with the aim of increasing production efficiency by exploiting the similarity of parts and their processes in assembly [Ham *et al*, 1985]. Basically, a number of machines are grouped into cells when they share a large proportion of operations they need to perform on a set of parts. If machines $m1$, $m2$, and $m3$ all need to apply their processes to, say, 80% of all parts, it would be advantageous to place them together in a cell and route the relevant parts through that cell. An overwhelming majority of these studies are based on routing information depicted in a zero-one, binary machine-component incidence matrix (MCIM) A , where

$$A = [a_{ij}]$$

$$a_{ij} = \begin{cases} 1 & \text{if component } j \text{ visits machine } i \\ 0 & \text{otherwise} \end{cases}$$

The objective is to minimise inter-cell travel time and maximise cell independence, but effective use of GT to form cellular layouts has enabled manufacturers to benefit from several other advantages over conventional batch or line production techniques. Reduction of set-up times and costs, simplification of material flow and handling, and standardisation of production processes are all derived advantages, depending on the environment. The most direct net result of its application is a mass-production effect (more constant, fluid flow) on any multi-product, small lot-sized production.

There are many examples of the application of GT to the facility layout problem. Initial approaches devised machine-part matrices and routed components along logical processing paths [Kusiak and Cheng, 1991]. When an appreciation of the advantages of a physically close location of similar machines arrived, more sophisticated algorithms emerged, including machine cell grouping and component routing, together with inventory costing and material handling (Askin and Chiu, 1990). Later approaches incorporated finer detail such as individual cell layout and material entry-exit points (Rajasekharan et al, 1998). Many others exist, but in essence, the main focus of work can be summarised as follows:

1. Group all parts which require similar processing into logical groups;
2. Create cells with machines corresponding to the processing requirements of those groups;
3. Locate those cells according to the least extensive inter-cell travel distance, taking into account cell entry/exit points;
4. Create a master routing path for the components according to processing requirements through the cells;

However, current efforts in this area ignore one or more of several issues, such as to facilitate within a budget, or to minimise makespan, minimise space (area required), maximise cell utilisation, incorporate process precedence (operational priority), allow for inter-cell travel times, production volume changes, machine capacity and processing rates, etc. [Jayakrishnan, and Narendran, 1999]. All of these factors have to be taken into account when designing a facility layout due to their interactions. For example, the number, type and position of

processes (or machines) within a cell or group of cells not only affects the facility's cost, but also the space required and the makespan of the assemblies. Cell machine density (number per cell) affects utilisation. Routing is currently a fixed form of scheduling, while in practice it should be dynamic in order to maximise cell utilisation. In addition, GT alone cannot easily cope with new assembly introduction, as this may require the complete re-designing of the layout, with catastrophic consequences if the new design is substantially different from the original. In summary, most methods based on a binary MCIM do not address the following issues:

- Sequencing of manufacturing operations, especially that of multiple visits by a component to a specific machine type;
- The effect on facility of introducing new models with similar parts but different processing requirements or combinations;
- Most methods require an initial value setting the upper bound on the number of machines within cells, or cells within the facility;
- Multiple objective layout design.

1.5.3 Proprietary heuristics

A great deal of effort has been given towards the minimisation of facility cost and intercell movement of parts. Beaulieu *et al* (1996) devised a two-stage heuristic for cell formation, using group technology in the first stage and intercell flow analysis in the second to eliminate under-utilised cells. A novel multi-function machine similarity coefficient helps in minimising total costs. This approach does not strictly apply to vehicle manufacturing, as multi-function machines are unusual in the BIW assembly line, where single-process robots tend to dominate. The incorporation of multi-process tool changers may revalidate the concept. However, no indication is given of intercell transport times, the problem size has been kept relatively small, and final plant layout details are not shown.

Lee and Chen (1997) propose a multi-criteria weighted approach for cell formation considering demand, batch and pallet sizes, routing, processing times, machine capacities and their workload status, giving one of the most comprehensive algorithms documented to date. The objective is to minimise intercell travel within the set constraints. Since a multi-constrained problem is difficult to optimise (creating a non-linear function) due to potentially contradictory effects, each constraint is given an importance weighting of between 0 and 1, set by the user. The algorithm uses a 3-phase approach and allows for duplicate (identical) machines within

cells. Machine-cell part families are constructed, then intercell movements are estimated and finally a cell formation is created. Improvement iterations are based on four sequential rounds of machine and part exchange and/or reassignment; their optimality is measured in terms of average intercellular distance. The benefit of this approach is a relatively short computational time, which is polynomial, even for the large problem of the given example of 60 machine and 180 part types. However, noticeably absent from the algorithm are considerations of facility cost, actual plant layout and the ability to mix dissimilar machines within cells, all of which are essential features for the design of agile facilities.

Jayakrishnan and Narendran (1998) developed a weighted clustering algorithm allowing for process sequence data, product volumes, and intercell movements. CASE (Clustering Algorithm for Sequence Engineering data) is a non-hierarchical clustering system where machine similarity is measured by the degree to which it shares parts with other machines. Unique machines (i.e. each resembling no other type) act as the 'seeds' around which other dissimilar machines (centroids) cluster, creating the basis of cells. A threshold affinity level (TAL) is set for machines, with totally dissimilar machines having a TAL value of zero. Parts are then assigned machine cells according to the minimum intercell travel times. Subsequent iterations improve the result by looking for higher TAL values, thereby identifying more centroids and adding machines to cells until no more changes can be done. Extremely rapid facilitisation can be achieved, and the authors claim the achievement of superior solutions for 20 part/8 machine problems in as little as 3 iterations. However, as with previous efforts, the algorithm ignores several important issues such as overall facility cost, travel times, physical layout, etc.

In general, proprietary heuristics may provide fast solutions only to a limited subset of problems. They tend to be subjective if used with weighted objective functions, and are difficult to apply to the complexities of the real-life industrial environment.

1.5.4 Genetic Algorithms

Another powerful approach is the use of Genetic Algorithms (GAs), which mimic the natural selection system to iteratively generate better solutions to layout and a variety of other problems. While optimality cannot be guaranteed, and for a large problem remains difficult, for small problems optimality it is quite feasible. An advantage of this system over numerical

methods is the ability to incorporate multiple constraints as well as non-linear functions. Traditional GAs broadly follow three sequential steps [Rao et al, 1999]:

1. A population is generated from the available data and the fitness of each individual calculated according to pre-set criteria;
2. A selection is done of the adults which will produce the offspring in the next generation, and mating is allowed to occur; random mutations may also be allowed to avoid premature convergence to local optima;
3. The offspring from next generation are ranked (for fitness) and the adults involved for mating in the next generation are selected; the cycle continues.

The set variables that are required are the size of the initial population, the setting of the fitness function, the probability of crossover during mating, the probability of mutation happening, the number of generations required for reproduction, and the upper bound on the number of machines in each cell. The fitness function is unique for each problem domain as no two problems are ever the same.

There are numerous examples in literature, and some of the more important are listed below. Conway and Venkataramanan (1994), increased the scope to the problem of changing facility requirements over time, minimising cell rearrangement costs by minimising interdepartmental material movement. Welgama and Gibson (1996) provide a more complete algorithm for cell formation, including cell layout, machine orientation and utilisation, MHS costs and the space used. Rules are used for determining the best MHS according to the type of material to be transported. Interestingly, aisle space is considered, and penalised, in an area optimisation attempt, useful in industry where new production lines are to be installed into existing plant space. Machine selection is done by considering total machine usage and distance between neighbouring machines. However, process sequences, makespan and cell utilisation are not taken into account.

Rajasekharan, Peters and Yang (1998) used GAs to design an FMS facility including cell shape, orientation and pickup/drop-off points, and concluded that their heuristic only took 1% of the time of a comparable author (Das, 1993), achieving optimal solutions for problems of up to 6 cells, and good ones for larger problems.

Rao et al (1999) utilised GAs to redesign an existing facility. Their objective was to minimise intra- and inter-cell material handling distance, a non-value added activity. They imported existing designs from AutoCAD in IGES format and created chromosomes based on those existing cells. Following the usual GA mating and replacement routines, they stopped at 750 generations and generated 10 layouts, varying the number of machines within cells from 3 to 9. The results were then re-imported into AutoCAD and simulated for operational data acquisition. The selection of the best layout was found with subsequent evaluation of machine utilisation, machine idle time, and a manual calculation of the costs for machine translocation. This itself was based on the original distance between cells and a 'changeover difficulty' value. As these factors were not used in the original redesign effort, the subjectivity of this method means further research is required.

The above shows that GAs can be successfully used for facility design, their problem type is very small. Most industry examples are much larger problems. Also, the authors concentrated on minimising the distance between the cells using a cost function that includes traffic density between the cells. This assumes that:

- A schedule is already known, and
- The cells themselves have already been generated, with their contents known.

Additionally, for anything but simple problems GAs are computationally slow and do not guarantee optimality. The success of these solutions is subjective, depending on whether a solution is deemed appropriate to the problem in hand, which in turn depends on how many iterations are done. Further difficulties reside in the selection of parameters that make up the 'chromosome', and how the 'fitness' value is determined. GAs should only be used in cases where optimality is easily defined, near-optimality will suffice, and time is not an issue.

1.5.5 Tabu Search

Tabu Search is a powerful methodology. This technique attempts to arrive at optimal solutions by generating an initial solution, then iteratively improving it until there are no more possible improvements (Chiang and Kouvelis, 1996). In order to avoid repeating past solutions or getting trapped into a local optimum, the system holds a list of past 'Tabu', or forbidden, solutions found. Some high quality solutions can be found in this way, but it does suffer from two drawbacks: not knowing the number of iterations required, and the possibility that, in a large or heavily constrained problem, it may be unable to find any feasible solutions at all.

Chittratanawat and Noble (1999) developed a multi-step heuristic for the simultaneous design of facility layouts, including pickup/drop-off locations and MHS equipment selection. The heuristic is based on a non-linear mixed integer program, structured as a Tabu search meta-heuristic procedure, with the objective of minimising facility cost. The procedure is different from the normal Tabu search algorithms in that it did not incorporate long-term moves memory, and used an initial layout construction algorithm for search initiation. Handling costs are based on equipment rather than distance and both distance, and material flow rate were used as dynamic variables. Trials on four 16-cell problems run for 5 hours (300 generations) yielded solutions on average 3.7% to 12% better (less expensive) than randomly generated solutions. The problems with this type of solution seeking are the restricted moves search diversification due to the memory-less system, reliance on the generation of an adequate initial solution, and the inability to evaluate the quality of a solution respective of an optimal value.

1.5.6 Simulated Annealing

Simulated Annealing (SA) is a random search algorithm also based on iterative improvement (McMullen and Frazier, 1988). It attempts to avoid being trapped in local optima in a search for the global optimum by accepting inferior solutions according to certain pre-set criteria. It takes its name from the physical annealing of solids, in which they are heated to a very high temperature and then cooled at a very slow rate, allowing the greatest amount of time when very near the freezing point of the solid. The purpose of this is for the crystals in the solid to have time to rearrange themselves, usually to acquire a desirable attribute such as strength or hardness. In combinatorial problems, an analogous process is used where an initial feasible solution is generated and an aspect is then randomly changed. In cellular terms this is equivalent to a cell accepting a new machine, exchanging a machine with another neighbouring cell, or generating a new cell altogether. The new solution is evaluated and compared to the objective function. This solution is then either accepted, if better than the initial one, or it is treated probabilistically, resulting in being accepted only if a certain acceptance criterion is met, when it becomes the 'current' solution. This enables SA to avoid converging into a localised optimum, and high quality (near global optimum) solutions have also been found.

Several authors have successfully implemented SA techniques for the cell formation issue of group technology. Proth and Vernadat (1991) were among the first to use SA for

manufacturing layout design. Their method was generation of the shortest path between cells as the objective, calculated by a branch-and-bound approach. As the shortest path is zero, occurring when all of the machines are present in one cell, cell size limits were imposed. Conscious of the many factors in cell shape, size, input and output points and its positioning, the two-phase algorithm was first used as an expert system (using rules) to find the best layout configuration type for the particular problem type, and then applied as an optimisation algorithm for the shortest path layout generation. This approach has the advantage of being able to deliver a precise physical cell layout with cell content and distance between cells. An example using 30 cells (870 paths) took less than 7 seconds. The limited nature of this early algorithm is the absence of many of the criteria important in facility design, such as cost, cell parallelism, part travel times and scheduling. Nevertheless, it is one of the few methods to give an actual layout.

Sofianopoulou (1997) combined SA with linear programming to generate good solutions to randomly generated problems. The objective was to create the smallest possible layout to minimise intercellular moves, setting size and process sequence as constraints cell, both of which are useful in industry. The results, based on a previously published problem as well as several randomly-generated ones, showed optimality was reached in most of them, and where this had not occurred, the best solution was within 3% of the optimal one. Solutions for 5-process cell size problems took between 57 minutes and eight hours to find. However, the author did not consider many essential aspects such as cost, MHS travel times or scheduling, which would have added considerable constraints.

Vakharia and Chang (1997) formulated the use of simulated annealing and Tabu search within group technology, generating near-optimal solutions. They consider parts, operations, machine types and cells, attempting to minimise procurement and intercell batch transport costs. Maximum cell utilisation and cell sizes are used as constraints. However, their heuristic does not consider minimum values for utilisation, area (space) constraints, MHS costs or any scheduling. As with Beaulieu *et al* (1996), final plant layout details are not shown.

One of the recent and more interesting attempts has been by Lee and Chen (1997), who consider cycle demand, batch sizes, pallet size, routing sequences, processing times, and machine capacities for the design of cellular facilities. Importantly, a large problem is used in the example and solved in minutes, making this approach more in line with today's

requirements in industry. A three-stage procedure is used; A machine-merging exercise following cell formation evaluation within a second-stage work balancing loop eventually leads to a viable solution, which is then optimised by a machine or part reassignment procedure. However, their multi-criteria method uses a subjective weighting technique that considerably influences the end result. More importantly, their third and final optimisation step is not iterative, thus optimal solutions cannot be guaranteed and no indication is given as to how the final solution could be judged. Machine processing capacities are estimated, as opposed to calculated, which may lead to inaccuracy. Finally, although in the workload processing time estimate, intra-cell parts movement is assumed to be negligible enough to be zero, no consideration is made of the more significant intercell travel time.

Su and Hsu (1998) applied a modified form of simulated annealing to the machine-part cell formation (MPCF) process with good results. Their aim was to minimise total intercell and intracell costs as well as intracell machine loading and imbalance. Their approach was to use a hybrid SA/GA algorithm, the main advantage being a much shorter execution time. The GA was used to circumvent the problem of initial feasible solution generation, and then SA's probability rules were applied in the cooling process. The example used came from previous literature problems and the authors claim that a superior solution can be reached, in a shorter space of time. However, no provision is made for intercell travel times or scheduling, plant layout is limited to two formats only and the problem size has been kept small (5 cells).

McMullen and Frazier (1998) utilised an eight-stage SA technique to solve the assembly line balancing problem for parallel workstations (cells) in a multi-product, mixed-model sequence environment. Seven experiments were run with previously published data; the two main objectives being to reduce the average cycle time and the total design cost. Several composite functions were created to moderate their objective function, using weights for average lateness and design cost. Initial settings for iterations, cooling rate and acceptable solution assessment were modified from previous authors. The time taken to find the best solutions varied from 1 minute for small jobs (11) to 20 minutes for the larger ones (74 jobs). Results showed that while their solutions bettered previous work in the former, only average solutions were found in terms of design cost. As with previous efforts, several factors (intercell travel times, scheduling, etc) were not taken into account. The above results have acceptable solution periods, but the problem sizes have been kept small. With higher cell and job numbers, solutions times would increase exponentially. In general, the main criticisms aimed at SA

techniques are their long execution time (due to the slow cooling rates necessary to improve the solution as it approaches optimality), and the fact that it is not a simple task to determine the best initial settings. A great deal of initial tinkering with variables such as the cooling temperature, the acceptance criterion, the number of iterations allowed and the number of changes per iteration allowed to occur (cooling rate) is required, making this a subjective as well as time-consuming exercise. Finally, it has been shown that it is not always possible to generate a first feasible solution. The severity of this problem appears to increase with the number of input variables and constraints imposed, making it less than ideal for use in a realistic FLP scenario.

1.5.7 Networks

Many optimisation problems can easily be defined as graphical or network representations because of their relational nature. A network represents a field of points joined by links, termed nodes and arcs. The analogy between networks and cellular facilities is not difficult to see. As part flow between cells can be quantified, networks easily lend themselves to the classical FLP such as shortest path problems (SPP), maximum flow problems (MFP), minimum spanning tree problems (MSTP), etc. The chief weaknesses of network models are the simplistic approach and exact nature, effectively correlating solution time with problem size – NP complete. Although useful for scheduling, routing and capacity requirement problems, they have limited use in industry due to the difficulty in correctly incorporating flow direction between nodes, equivalent to allowing unrestricted and polydirectional movement between any 2 cells. Flow in the space between cells, such as intercell aisle space, cannot be modelled (Irani, Cohen and Cavalier, 1991). Nodes may represent machines or cells, but multi-machine cells such as those assumed in this work have traditionally been difficult to formulate. Finally, important issues such as physical layout and cost have not been incorporated.

Recently Wu (1998) devised a network-based method for the design of multi-machine cellular layouts. Apart from the previously used simple (one machine) nodes, complex nodes with more than one identical machine have been devised, a useful attribute in any future agile facility design. Starting from capacity requirements, cell formation rules guide the algorithm in six stages to generate a machine-cell complex that describes the least number of intercell movements. Two examples of 6 parts/8 machines and 13 parts/13 machine types are given, where the author claims good results are arrived at. However, this approach, although more

developed than other network examples, suffers from many of the characteristic problems of using networks: no final layout can be arrived at, there is no consideration of the design costs incurred, and problem size.

Similar to genetic algorithms in scope but not in context, neural networks (NN) are algorithms that attempt to mimic nature. Information is stored and created in the form of neurons, and the greater the number of times a specific information unit is passed between two neurons, the more reinforced is the path, and hence the 'memory'. Associating the best 'neuronal layout' with the highest travelled signals develops solutions. Unlike other artificial intelligence (AI) methods such as expert systems (ES), whose characteristics include sequential information processing, explicit knowledge representation, and use of deductive reasoning techniques, NNs utilise a parallel approach, possess implicit knowledge of systems and apply inductive reasoning (Hao, Shang, and Vargas, 1995). Such parallelism overcomes the conventional If-Then rule approach of most other approaches, the advantages mainly being (theoretically) much faster solution times and greater flexibility. In reality, the difficulty in devising a realistic model, the directional control of solution development and especially, the high CPU processing power required can lead to long solution times. Some authors have reported difficulty in finding global optimal solutions for combinatorial problems such as the FLP.

1.6 Requirements of the necessary methodology

The literature research and analysis in chapter 1 reveals that there is no single best method to solve the facility layout problem. Each method has its advantages and disadvantages and some authors (Huntley and Brown, 1991, Tanaka and Yoshimoto, 1993, Gau and Meller, 1999) have attempted to combine them with mixed results. Facility design is a complex problem where a multitude of production factors may need to be evaluated and incorporated into the design algorithm at an early stage, and many authors have chosen to approach the problem sequentially rather than simultaneously. However, most of these factors are inter-related and not considering one will inevitably result in an incomplete solution or at best, a partially improved one being generated.

What is clearly required is a methodology that incorporates the best of the available methods for each necessary step, but is designed to utilise all of them simultaneously in the search for a solution. Following the review of the existing literature, it has become apparent that there are

at least 10 criteria required to generate good quality solutions for agile facilities within a reasonable time, insofar as their use in industry is concerned:

No.	Criterion	Reasoning/comment
1	Problem-specificity	It should be related as closely as possible to the problem type or category to be solved, as the usefulness of generalist methods is limited;
2	Initial search direction method	The methodology should begin with a guided attempt at an initial layout creation and not rely on random designs; the solution-searching methods which rely on random starting points are more likely to either reach local optima and/or take longer to reach 'acceptable' solutions;
3	Solution improvement guidance method	Given the complexity of the problem, the solution improvement method should not be single-pass, but incremental or iterative. The latter is the best method as it allows a layout to be evaluated before being accepted or rejected;
4	Solution cycling avoidance	The iterative changes made and/or resulting layouts (solutions) generated should be either stored for future reference, or somehow not made to reappear more than once; such repetition is wasted effort and may lead to local optima;
5	Visual layout format	The methodology should be able to design a cellular layout, specifying cell contents in terms of number and type of processing machines, entry and exit points and size of layout;
6	Goal orientation	The methodology should be goal-oriented in the form of being objective-driven. Given the complexity of requirements there may be more than one such objective and these should be limited by constraints, as they are in real life;
7	Goal achievement	The methodology should generate industrially acceptable solutions and, where known, not too distant from the 'global' optimum for the problem;
8	Solution finding speed	The methodology should be reasonably fast even for problems above the known optimality limit of 12 machines;
9	Presence of scheduling	The methodology should incorporate scheduling aspects;
10	Methodology implementation	The methodology should be implementable into a visual program, which can be utilised and understood by industrial designers when designing such layouts.

Table 1.2 The 10 main criteria necessary for speedy facility layout design that can be successful when used in industry.

1.7 Reasons for selecting the proposed methodology

The emerging necessity now is to determine which of the existing methods, if any, may be used to fulfil these 10 points. It is worth separating the problem into two domains, that of layout design and that of scheduling. However, the principles of search apply to both. Each search method has advantages and drawbacks, and they can be compared graphically by

plotting the probability of each method of finding the global maximum (or minimum) against size of solution space.

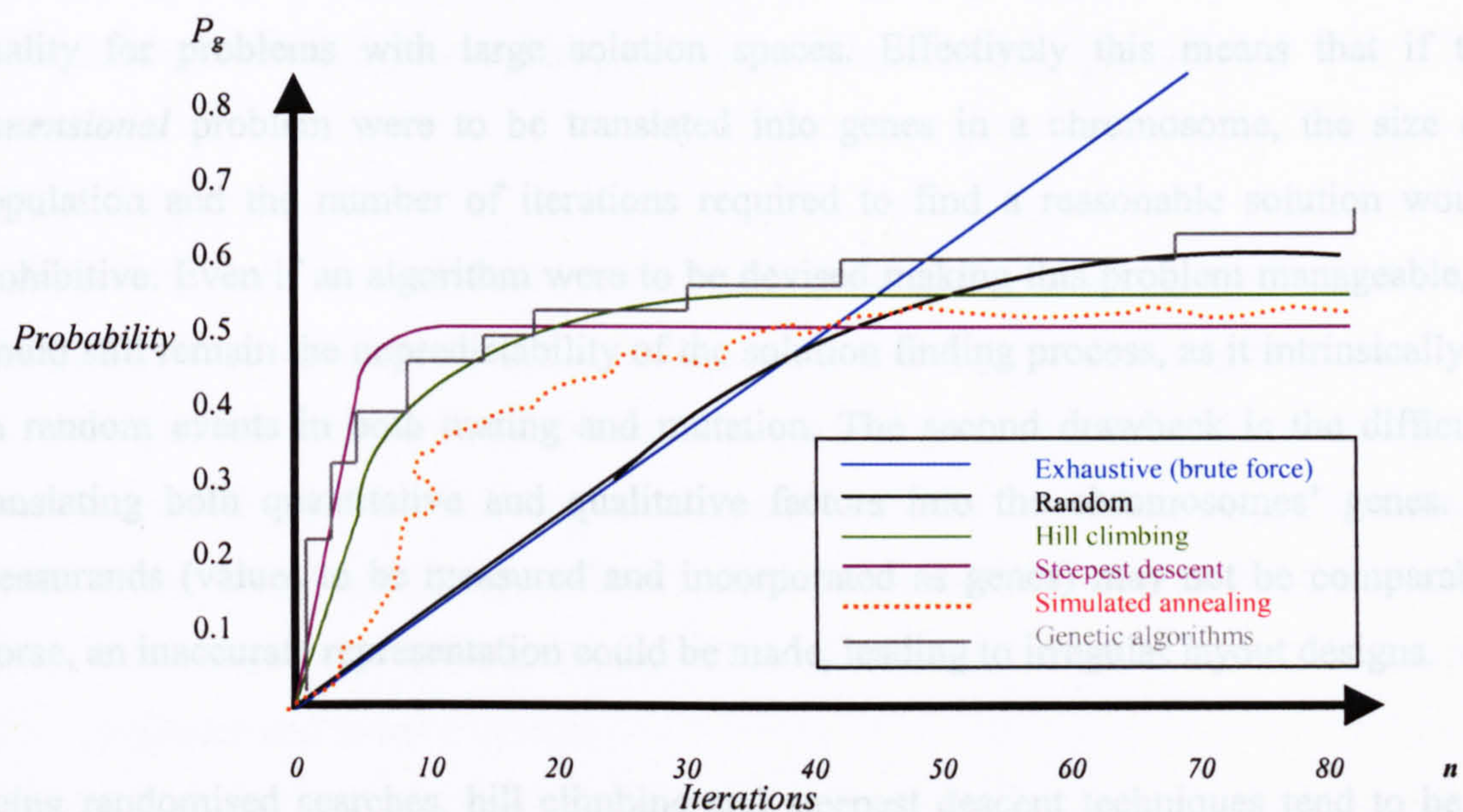


Fig. 1.5 The probability of various search methods of finding the global minimum/maximum with number of iterations; the path for simulated annealing should be seen as randomly variable; genetic algorithms improve rapidly to start with but time between improvements subsequently gets larger, though they do not stop due to mutation [Johnson and Picton].

1.7.1 Analysis of best method for layout design

A methodology for efficient layout design should consist of one or more of the above methods incorporating those principles that best address the issues of multi-factor complexity and the corresponding size of the solution space. As optimality is not the aim, the methodology should try to achieve a satisfactory solution, meeting the set objectives as well as satisfying the required constraints, ideally within as few iterations as possible.

1.7.2 Concept justification for SIMAID

Given this scenario, this rules out the exhaustive and random search methods. Utilising simulated annealing algorithms may result in unpredictable search tangents and a varying quality of solutions, leaving the genetic algorithms, hill climbing and steepest descent methods. These tend to approach the problem at approximately similar search rates, are awkward to work with when dealing with multi-criteria optimisation, and fall into the exponential (polynomial) problem size category, so other, faster ways must clearly be found.

Genetic algorithms, unlike simulated annealing techniques, can sustain the solution improvement process almost indefinitely, but suffer from two main drawbacks when used to generate layouts. Firstly, like SAs they cannot be relied upon to provide consistent solution quality for problems with large solution spaces. Effectively this means that if the *n-dimensional* problem were to be translated into genes in a chromosome, the size of the population and the number of iterations required to find a reasonable solution would be prohibitive. Even if an algorithm were to be devised making this problem manageable, there would still remain the unpredictability of the solution finding process, as it intrinsically relies on random events in both mating and mutation. The second drawback is the difficulty in translating both quantitative and qualitative factors into the chromosomes' genes. Many measurands (values to be measured and incorporated as genes) may not be comparable, or worse, an inaccurate representation could be made, leading to irregular layout designs.

Being randomised searches, hill climbing and steepest descent techniques tend to be faster than any of the above but tend to fall into local minima. As shown in fig. 1.4, they also rarely rise above 0.5 p for finding the global minimum. Hence, neither of these methods appears to be suitable for the purpose.

As single methods do not appear to be adequate on their own, a combination may prove more successful. Combining the fastest method, steepest descent, with a 'safety net' to allow the algorithm to avoid the local minima trap appears to be a better approach. Further, if the methodology exhibits some of the qualities of genetic algorithms, such as the ability to keep improving a solution over time, then it will be able to satisfy the requirement of solution selection by the user.

1.7.2 Concept justification for SIMAID

The program designed from this methodology is called SIMAID (SIMulation AID). This has been designed with the above concerns in mind, and thus built in three iterative stages, a guided, initial layout generating stage and two, recursive, solution improvement ones. For each stage, a different technology is utilised: Either an existing technique, a modification of an existing technique, or a novel one not documented previously.

Iterative layout design methodologies have recently begun to appear in literature, usually as a mix of two or more techniques. An indicative method, using alternating mixed-integer

programming (MIP) and genetic algorithms, formulates the problem as a MIP and then uses GAs to overcome the problem size limitations of MIP (Gau and Meller, 1999). The authors reported an average improvement of 10% over previously published (single-technique) solutions to problems, indicating the potential usefulness of this approach.

1.7.2.1 The first stage

This method bears some similarity with CASE (Nair and Narendran, 1998) Like CASE, it was designed to overcome two major failings in layout design methodologies to date:

1. Algorithms based on hierarchical clustering have the severe limitation of irreversibility – that is, processes clustered into cells cannot be changed later; and
2. Machine aggregation into cells is carried out not only by the classical GT measures of cell compactness and the minimisation of inter-cellular moves, but also on process sequence and volume criteria.

The usefulness of incorporating production and volume data for cell formation has been noted before (Seifoddini and Djassemi, 1995). This increases the chances of components with high production volumes being processed within a single cell (thus limiting the number of intercellular moves), and also avoids the necessity of incorporating separate capacity planning and cell size limitation functions. Cells can thus be designed by actual production requirements, and not by artificial aspiration criteria.

SIMAID's first stage generates an initial (infeasible) layout solution. Because this is a process-driven (i.e. based on actual production process requirement) and not a random method, the solution tends to be in the area of the solution space in the proximity of the global minimum. The cell-making algorithm is process sequence-based, and generates a first cell layout by aggregating processes into cells depending on type and quantity (based on volume) of components which are likely to visit them. Like CASE, it uses a cell 'seeding' procedure to create the cell nuclei, from which whole cells are formed. Unlike CASE, however, it does not use any (dis-) similarity coefficients or proximity measures, which can be subjective and, importantly, may invalidate one of the aims of agile manufacturing, facility reconfigurability. Unlike both CASE and other GT methods, it does not group processes into cells based on part families, but on subassembly process sequence. Likewise, it does not attempt to evaluate the derived solutions by using a 'meaningful criterion'. SIMAID evaluates solutions by simulating the layout under operational conditions.

1.7.2.2 *The second stage*

The second stage includes scheduling, where the initial and subsequent layouts are used to create a good schedule for the subassemblies to be processed. Scheduling draws different requirements to layout design, as it can be both more simple and intractable. Being a sequence of subassemblies that need processing, it is a far simpler problem than layout design, where many factors contribute. However, schedules may be huge; hence its potential intractability (see chapter 3, section 3.41 for a description of the problem). Any methodology used must not only be fast, but also be able to reach very good solutions, at or very near the optimum. In order to do this a continually improving algorithm would be useful, as would a means of avoiding getting trapped in local minima. Genetic algorithms (GAs) not only fulfil these criteria, but also lend themselves well to the sequential type of problems which scheduling falls into. However, even the most highly efficient GA cannot escape the fundamental issue of problem size. Hence, the GA-based methodology must be designed to limit the size of the solution space by indirect means.

Substantial literature exists on the use of GAs for scheduling purposes (Davis, L., 1985, Goldberg, D., 1989, Falkenauer, E., 1991, Gen, M., 1994, and Zhang, C., 1995). Originally applied to job shop sequencing, the uptake of GAs has been slow due to the twin issues of finding a suitable gene encoding system, and the dimensionality of the problem. The most common method, that of translating the allocation of jobs to machines into strings, has the severe limitation of forming preset routes, which, if designing a layout, is clearly not suitable. Also, unlike conventional production lines, parallel cell agile facilities, which may possess multiple identical machines or cells, do not favour such approaches. In addition, given normal production volumes in the order of several hundred vehicles per day, and the potential size of the respective facilities, computer-intensive methods like GAs would not be able to cope.

The use of GAs in scheduling problems needs, therefore, to be carefully controlled. The GA used with SIMAID, with schedules represented as chromosomes and the individual components as genes, circumvents these problems by the use of two techniques:

- a) Restricting the initial chromosome formation process to viable (not stillborn) sequences, and

- b) Limiting their size to a ‘sequence window’, proportional to the size of the facility.

The first technique has seen considerable implementation, in various forms, by several authors (such as Davis, L., and 1985, Goldberg, D., 1989), but the second is a novel technique and is intrinsically suitable to parallel cellular facilities, where multidimensionality and cellular duplication abound. For crossover, scheduling has notoriously been a difficult area due to the potential generation of illegal offspring under ‘normal’ GA rules. SIMAID uses a ‘sequence-extracted’ crossover operation, similar to that used by Wang and Brunn, (Wang, W. and Brunn, P., 2000), the main difference being the ability to create 4 offspring instead of just 2. Also similar to their work is gene mutation by neighbour swapping, but this work also introduces the novel concept of transposition. Finally, simulation is used to evaluate their fitness. Details are given in chapters 3 and 4.

1.7.2.3 The third stage

The third and final stage comprises a search made of three separate, cyclic phases, the first two of which are potentially recursive. This is an improvement stage that searches for the change that could be made to the layout in order to achieve the single biggest improvement within the iteration. Graphically, this could be described as the steepest descent, as this represents the fastest way down the ‘problem hill’ towards the solution. Steepest descent, a non-linear programming technique, has been used for optimisation problems for decades (Nicholson, T., 1971), but it has not been popular for layout design due to its tendency, when unaided, to fall into local minima. This stage is described in detail in chapter 4.

As implemented in SIMAID, the steepest descent method, although not guaranteed to reach the global minimum, tends to approach it avoiding the vast majority of local minima. This is because the initial layout design is devised by product sequence data; hence it’s unlikely that resulting solutions will be too distant from it. To minimise the possibilities of falling into a local minima, the third stage has been designed to cycle between the first two phases in a recursive fashion, if certain conditions are not met. Additionally, as in SA, infeasible solutions are also referenced for the generation of future solutions, allowing the search to explore other areas. A summary of the three phases is given below; details are given in chapters 3 and 4.

The first phase utilises a ‘dual factor’ steepest descent search method to find a viable local minimum. The dual aspect uses two defining criteria, cell request and process utilisation. The change that is implemented is that which would make the biggest reduction in makespan. Once this change is implemented, the next iteration begins. In order to avoid arriving to the same solution twice, a list of layouts already found is maintained – much like in Tabu methods. Unlike these, however, any changes to the facility layout are not the result of random (or probabilistic) methods, but specific, makespan shortening or cost-saving changes. Also unlike most Tabu techniques, the list contains *both* viable and infeasible solutions, enabling the search to avoid looping into repeating patterns.

The second phase alters the cellular layout by analysing each individual cell’s utilisation. If a layout has already been achieved (i.e. present in the Tabu list), then the second phase cycle stops and returns to the previous one. If not, it proceeds to the next iteration, recording the solution in its list. The third phase is invoked when better solutions cannot be found by the first two phases, attempting to reach the bottom of the minima (whether local, regional or global) by rearranging the cells and/or their contents by space and cost-cutting measures. Regional minima are defined as those that are inferior to the global minimum, but better than all the nearby local minima. By the end of the third phase, the layout design cannot improve any further by the methods used and SIMAID stops. The combination of these techniques is both novel and effective, and has not been seen in the literature to date.

1.7.2.3 Testing the methodology

Both the manner in which the research was conducted and the testing method are indicative of the problems faced with designing for a concept not yet in existence to date. Bearing in mind that the main aim was the creation of a methodology for the designing of cellular, *agile* production facilities, it proved difficult to source the test data as no such facility had been built, to the author’s knowledge, to date. The closest approximation had to therefore be sought, which was forwarded by a SALVO programme collaborating company, Lamb Technicon, in the form of production plans for existing assembly facilities. The latter had been manually configured and had conventional layouts, but had been extensively refined and “optimised” using commercial discrete event simulation software. The three layouts were then compared in terms of process count for the same expected production level.

1.8 Conclusion

It is evident that what is required is a comprehensive approach to the problem, with as many factors included in the design stage as feasible, without making the problem intractable. No single, exact method is possible but a sound methodology that incorporates several such tools provides a solution, heuristically implemented into an algorithm. A methodology that incorporates the objectives of cost minimisation, production flexibility (or agility) and facility utilisation maximisation has not yet been seen (Mansouri, S.A., *et al*, 2000). SIMAID has been devised with this in mind, giving the industrial facility designer a useful tool for layout design within a reasonable time span. The methodology was constructed starting from first production principles with raw data, using leading layout design technologies adapted for agile principles, and incorporated into loosely linked prototype code. This was subsequently integrated with databases and given a data-driven interface and a result window, creating a functional concept demonstration program.

This methodology is sufficiently flexible to allow for easy user parameter modification but is rigorous enough to allow the creation of optimal or good sub-optimal solutions, in a reasonable period of time. To achieve all this, it is necessary to begin by designing the boundaries of the problem by approaching it from first principles. Chapter 2 will explain the programme background that led to the definition of the requirements for this work. The subsequent chapters will describe the methodology itself, with an overview in chapter 3 and the principles in chapter 4. Chapter 5 will describe the 3 case studies and chapter 6 the conclusions.

Chapter 2

2.1 The SALVO programme

A recurring problem for vehicle manufacturers since the earliest days is the time and costs associated with retooling and re-jigging the lines if a model facelift is required, a new model needs to be introduced or even if a new process needs to be installed to replace an obsolete one. Nissan was faced with the prospect of spending \$500 million (about £356 million) in adding a new line for a model in a plant which was already producing two other models on two lines (Palmer, 2001). Such levels of investment cannot be sustained indefinitely and allow manufacturers to remain profitable, and Nissan found another solution in scheduling the new model on its existing lines. In this case Nissan was able to do this because the three models were fairly similar and shared many components between them. In the future, however, car companies may need to produce fairly dissimilar products in the same factory, where component commonality levels may not warrant similar approaches to production.

Part of the high levels of costs required for new model production stem from the tooling required for the pressing of the monocoque panels that make up the vehicle's body shell. Each new closure (door, bonnet or hatch) requires new tooling, as well as dedicated jigs to hold in during the production process. Also, the tooling wear rates tend to be high and on high-volume runs this tends to be often replaced. In order to avoid such a situation, recent interest has focussed on the use of aluminium spaceframes. A spaceframe vehicle differs from current types in that the main, load-bearing body structure is not made of pressed (stamped) steel sheets, as in current vehicles, but of extruded (in this case, aluminium) members, joined to each other in a variety of ways.

The scope of the SALVO project was to investigate the technologies required for the design of agile manufacturing facilities for aluminium spaceframe-based future vehicles, although the concept could be extended to any other type of vehicle, including current monocoques. Fig 2.1 below is an example of such a structure:

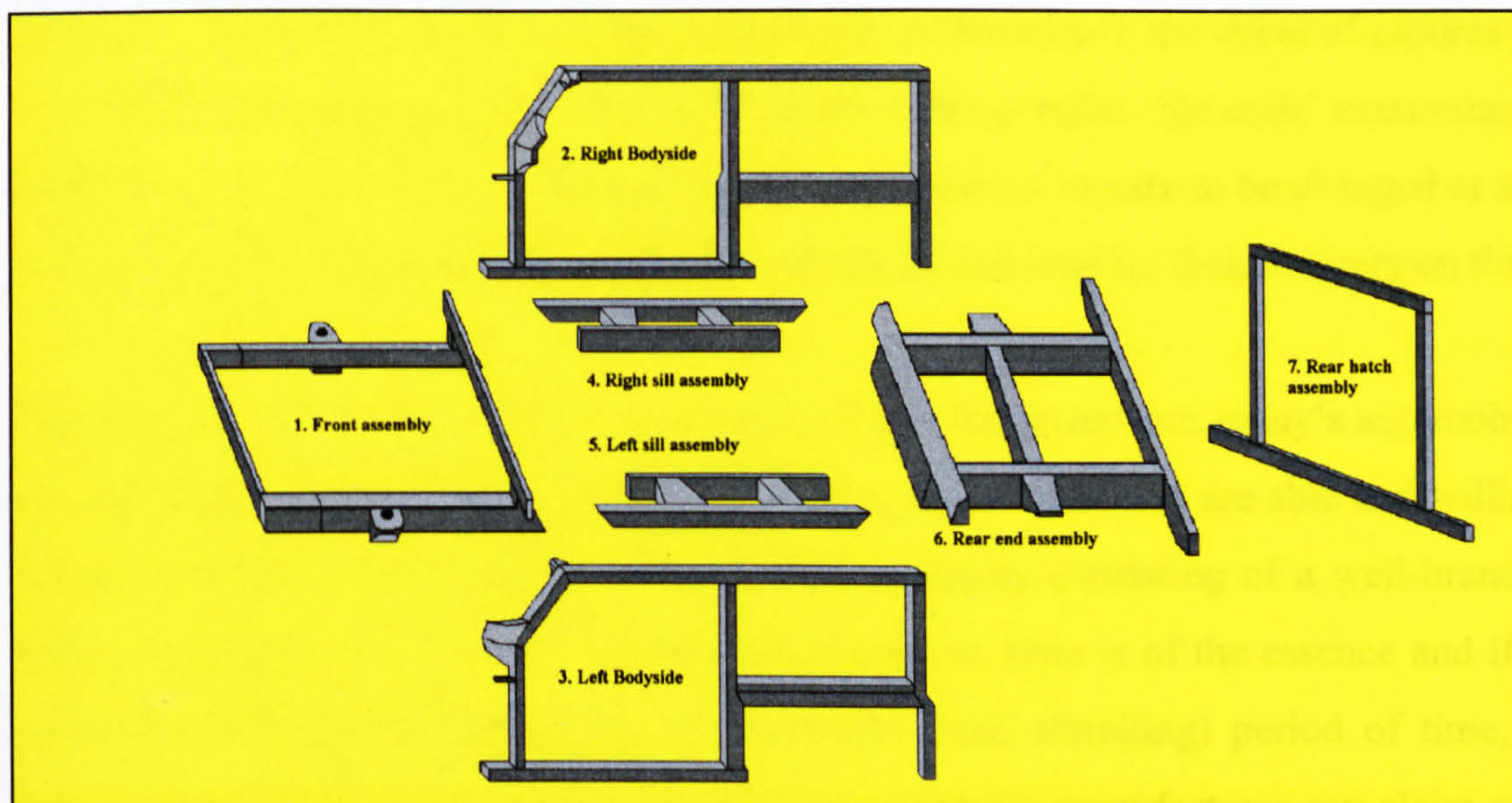


Fig 2.1 A possible example of a spaceframe vehicle body structure showing parts grouped (assembled) into distinct sub-assemblies.

Each of these subassemblies requires the joining of several parts together. As spaceframe parts tend to be joined with at least one end of one to any point of another, the variety of types of joint will inevitably be large. Combined with the differing roles to be played by each of these joints, this will have a knock-on effect on the types and number of processes required. A load-bearing joint in the forward sub-assembly which, for example, holds the engine bay, needs both strength and vibration resistance; a production engineer may designate a combination of two types of joining techniques for this joint, such as a mechanical fastener together with an adhesive application. An agile factory would need to assemble several such models and it can be seen that as structural requirements change so do the processing requirements. For late configuration purposes it may be advantageous to have as much commonality of parts between the different models as feasibly possible. This certainly applies to processes. Finally, to facilitate assembly a modular assembly structure would bring several benefits. It is extremely important that the vehicle models are designed with the assembly system in mind (i.e. design for assembly). Extrusions can be designed to form part of a family that are identical in shape apart from their lengths. Not only does this enable easier robot-fixture interaction but it provides further late configuration benefits as the model to be made does not have to be decided until immediately before the information is passed onto the local workstation which controls the cells (instead of weeks before).

In designing a facility, the cells' degree of automation will depend on the volumes and variety of vehicles to be assembled. However, as a guideline, if automated with several robots it

would be beneficial to build-in some operational redundancy in the event of failures as well as to be able to assemble the parts at a speed which does not reflect the cells' maximum potential. In other words, should any breakdowns occur or processes require to be changed or added, this will not significantly affect any production schedule, and enables their delivery on time.

One final point to consider is the importance of this last statement; today's automobile market is vast, varied and extremely competitive. Customers do exist that are able and willing to wait weeks or even months for a car of their choice, usually consisting of a well-branded model with a premium price. For the vast majority, however, time is of the essence and if a vehicle cannot be delivered to them within a reasonable (and shrinking) period of time, they will simply head for the competition. This is true as no single manufacturer can claim an absolute dominance over the others in terms of the values customers look for, such as technology, design, price, etc. There also seems to be a certain 'convergence' to certain design standards, resulting in a similarity in external design, for some models. One of the drivers for product differentiation and shorter time-to-market today is, in part, due to this.

2.2 *SIM*AID

In industry today vehicle manufacturers frequently find themselves having to add a new model or variant to their product range without the necessary funding for building a completely new, 'Greenfield' plant. More often than not, production planners need to find space in existing plants, often of reduced dimensions, and even of irregular layout. Existing equipment also may need to be recycled for use, and redundant employees retrained. As in any business today, achieving all this as rapidly as possible is a major desirable – thus enhancing the firm's competitiveness through an ever decreasing time-to-market. This principle also applies to many of their suppliers, such as the 'line builders', who bid for contracts to design and build the assembly lines for the major players. If the latter can turn bids around in days instead of months, their competitive advantage will increase. In order to do so, however, new working methods and tools must be adopted.

Currently, much of this work is done by a combination of a preliminary rough hand calculation, followed by discrete-event simulation, taking weeks if not months to produce a viable layout suitable for bid submission. Usually, hand and calculator do the initial background work from the process requirements and volume required, and then a first model is created on a simulation

package based on this. This model is then pushed through numerous manual improvement cycles until the planners feel they cannot improve it any further. While reasonable layouts may result, all of this may take months to do. Today, no methodology or program exists able to consider all the minutiae involved, and come up with an optimal (or sub-optimal but adequate) solution. Any model, however complex, will inevitably be a simplification, and its solution, an approximation. However, experience has shown that even an approximate (good sub-optimal) layout, achieved quickly, is a great leap forward, allowing the planners more time to concentrate on the details instead. It is in this role that SIMAID was envisaged.

2.2.1 The SIMAID objective

The program, SIMAID, stands for SIMulation AID. This is not related to the European Commission's ESPRIT programme in any way. The choice of acronym simply reflects the best interpretation of the function and aims of the program, which is meant as an aid to further (optimising) simulation effort. It is not meant to replace current commercial simulators in either scope or function. Its purpose is to enable the production planner/facility designer to create a 'good' facility design covering the layout and content of the cells, the facility dimensions and cost, and production information such as routing, throughput and scheduling. This can then be taken to a commercial simulator and modified as required. The scheduling may or may not determine the final subassemblies' routing but does contribute towards the cell layout design. In addition, other useful characteristics of the facility such as cell utilisation and breakdown resistance, routing flexibility, capacity flexibility and intercell traffic can be worked out, aiding in the selection of the appropriate MHS. All of this takes no more than a few hours, or a day at the most. The resulting model can be used as seen, or it can be exported to a discrete-event simulation package, and fine-tuned to desired specifications.

2.2.2 The SIMAID production philosophy

In today's environment car assembly plants are paced, with regular cycle times pre-defined and (usually) models produced in batches. However when using this system a great deal of flexibility, and corresponding agility, is forfeited. If thousands of customised vehicles have to be assembled and delivered within 3 days, batches by necessity have to be very small, tending towards the size of one. With 5 to 10 models per plant, the challenge can easily be understood. Although batching may seem necessary, it actually isn't, provided an adequate production control system is used within an agile facility. As the subassemblies will have different process

requirements their total processing times will be different. What this usually translates to is that the pace (cycle time) is dictated by the ‘slowest’ subassembly (that with the longest process requirements), and also that some line balancing has to be done.

In this work a different control system is assumed, that of cycle-less random cell access. This essentially means that the subassemblies are sent to whatever cell is available to take them (consistent with their processing requirements) and not in previously determined routes. This is a particularly important issue when considering real life process time variation, rework and cell breakdowns. The real-time controller is constantly monitoring the subassemblies’ progress and tells them when to move on to which cell.

2.2.3 Flow control

There are five main types of product flow [Kusiak and He, 1997]. Fig. 2.2 illustrates this below.

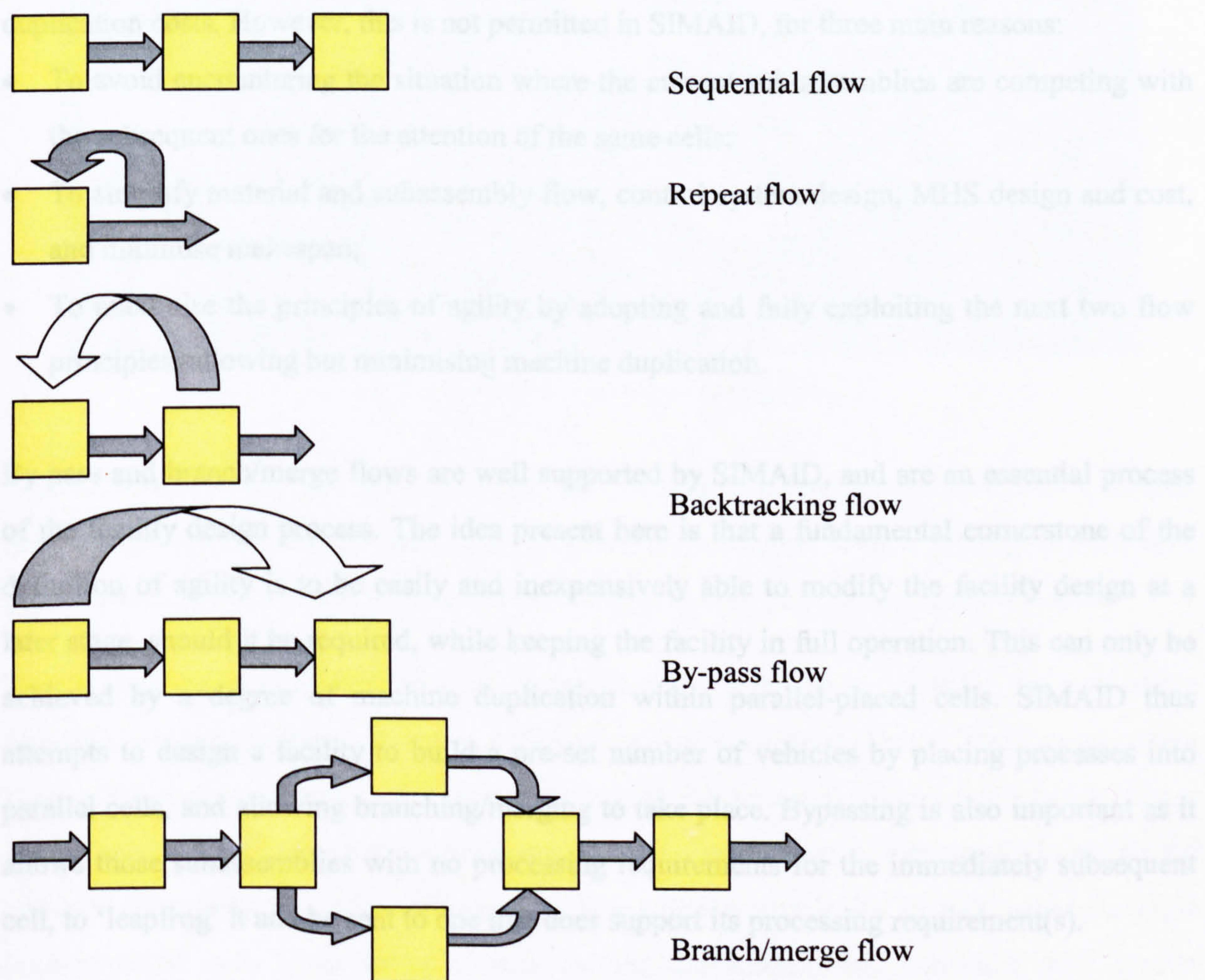


Fig 2.2 The 5 main types of part/subassembly flow, sequential/serial, repeat, backtracking, by-pass and branch/merge flow.

Sequential flow is the most common and logical, as there usually are some steps in any assembly/production process that need to be executed before others. This type of design is supported by SIMAID.

Repeat operations are usually allowed to occur when an operation is required to be done more than once. The main driver here is cost, that is, to avoid including a repeat cell or machine after the initial one. This does not apply in our case, as any process (understood as a robotic or manual production operation) is able to perform as many operations as required within a cell.

Backtracking to previous cells occurs when an earlier operation needs to be executed again after the current one, but before a third in the sequence. Again, this usually happens when planners wish to avoid including a repeat cell or machine after the initial second one, usually for cost reasons. This principle allows for greater flexibility and tends to reduce machine duplication costs. However, this is not permitted in SIMAID, for three main reasons:

- To avoid encountering the situation where the current sub-assemblies are competing with the subsequent ones for the attention of the same cells;
- To simplify material and subassembly flow, control system design, MHS design and cost, and minimise makespan;
- To maximise the principles of agility by adopting and fully exploiting the next two flow principles, allowing but minimising machine duplication.

By-pass and branch/merge flows are well supported by SIMAID, and are an essential process of the facility design process. The idea present here is that a fundamental cornerstone of the definition of agility is to be easily and inexpensively able to modify the facility design at a later stage, should it be required, while keeping the facility in full operation. This can only be achieved by a degree of machine duplication within parallel-placed cells. SIMAID thus attempts to design a facility to build a pre-set number of vehicles by placing processes into parallel cells, and allowing branching/merging to take place. Bypassing is also important as it allows those subassemblies with no processing requirements for the immediately subsequent cell, to 'leapfrog' it and be sent to one that does support its processing requirement(s).

2.2.4 The Tooling

The SALVO6 programme spawned several novel technologies to adapt spaceframe assembly for mass production. Unlike present day monocoque assembly techniques, which require sequential part loading for any assembly operation¹, spaceframe components are far more rigid and can be loaded onto a holding frame, or 'tooling', all at one time. Fig. 2.3 below shows the SALVO prototype tooling.

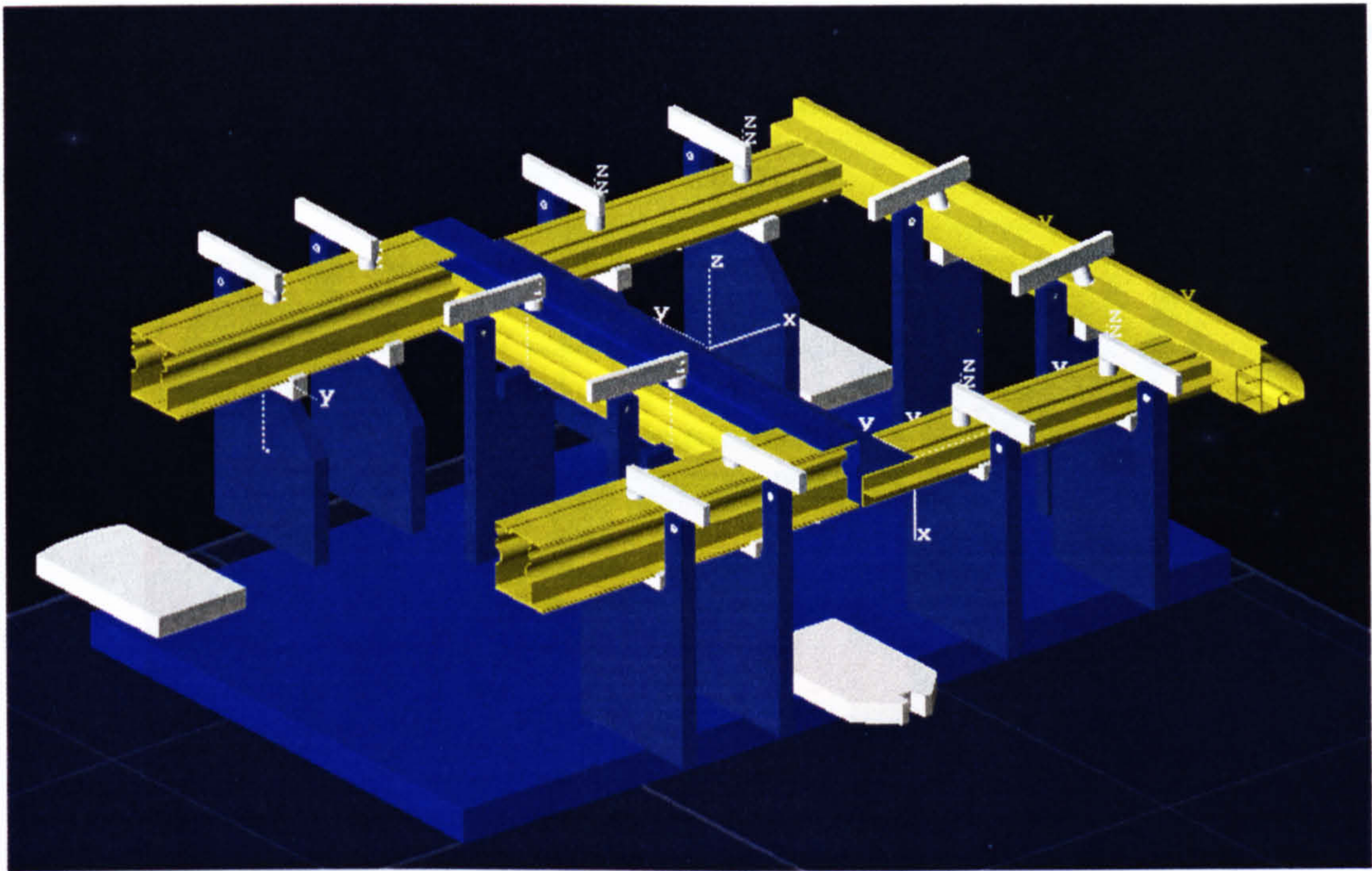


Fig 2.3 The SALVO prototype tooling pallet

The advantage of this arrangement is that part transfer is greatly diminished, and a simpler (and cheaper) MHS can be used. The programme developed such a tooling construction, placed it on a conveyor, and is therefore functioning as both parts holder and a transport.

This is ideal for an agile facility: most, if not all of the parts can be loaded at the start of the assembly process and firmly clamped with built-in wire-controlled clamps. The tooling is then shunted (by conveyor or any other means, such as an Automated Guided Vehicle (AGV)) from cell to cell until its processing requirements are completed, when it is channelled towards the unloader for putting into the framing station. The agility aspect is that by this means no single predetermined path needs to be chosen, leaving the subassembly controller with greater

¹ Monocoque structures are made of flexible sheets of metal, mostly steel. It is extremely difficult to hold these stampings in a fixed co-ordinate configuration in space without extensive, and cumbersome, clamping. Hence, the usual procedure is to load only the two parts that currently need assembly at any one time, firmly clamp them, join them and then proceed to the next operation.

freedom as to where to send individual tooling units. Should a particular cell be busy, the control system may send the tooling unit to another cell compatible with the processing requirements of the subassembly loaded onto it. Should a cell require maintenance, a process change, etc., the controller can be programmed to avoid that cell, but still maintain production through other operational cells in the facility. Should a new model be introduced with part of the requirements of the existing cell set-up, the controller can easily be programmed to send the new model to this existing cell, without disruption. In summary, the combination of spaceframe construction and a moveable tooling allows the principles of agile manufacturing to be earnestly applied. Fig. 2.4 shows the SALVO prototype production facility, and Fig. 2.5 shows a diagram of a hypothetical of an agile facility.

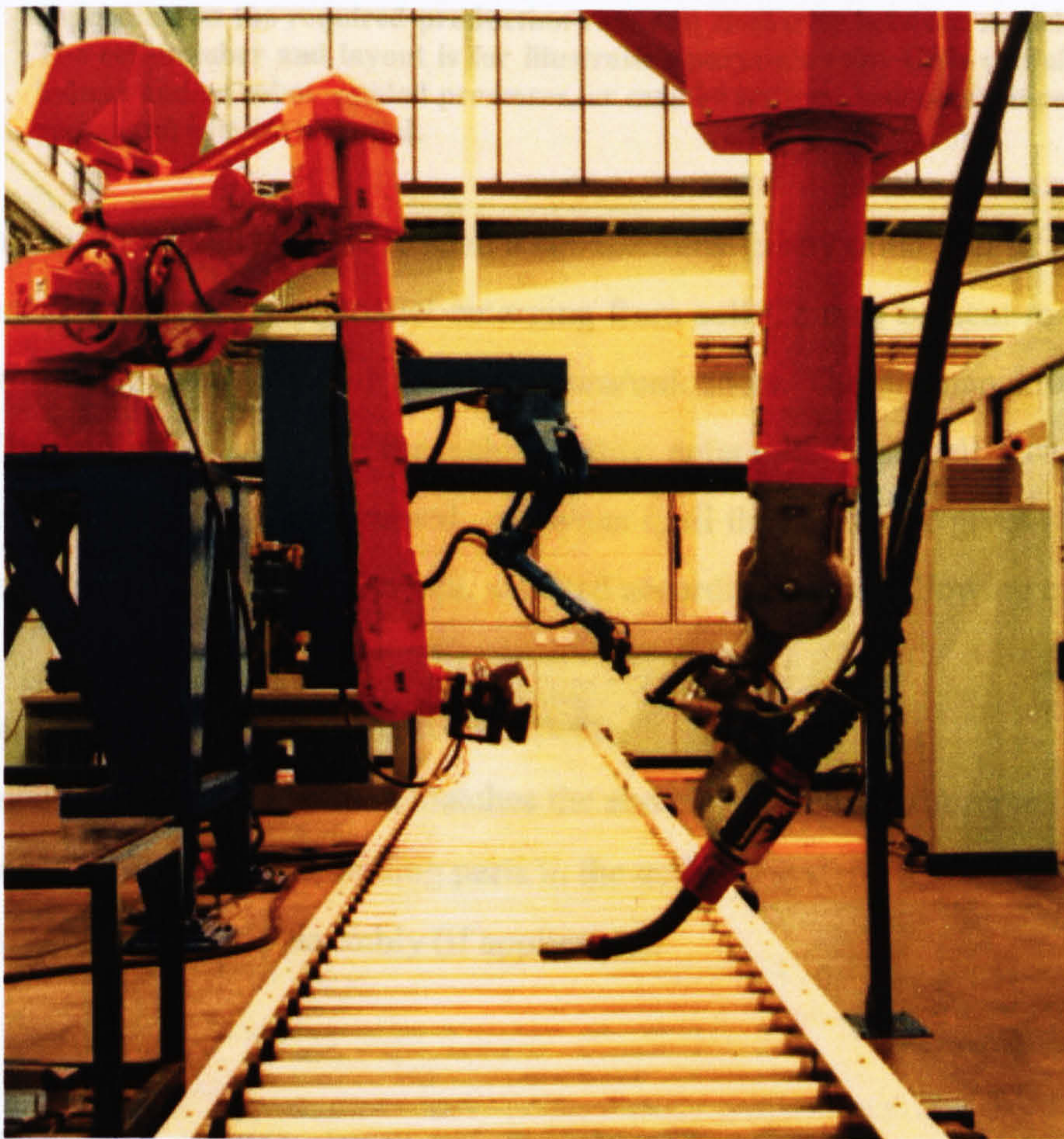


Fig 2.4 The SALVO prototype agile facility, revealing the three cells with distinct robotic processes.

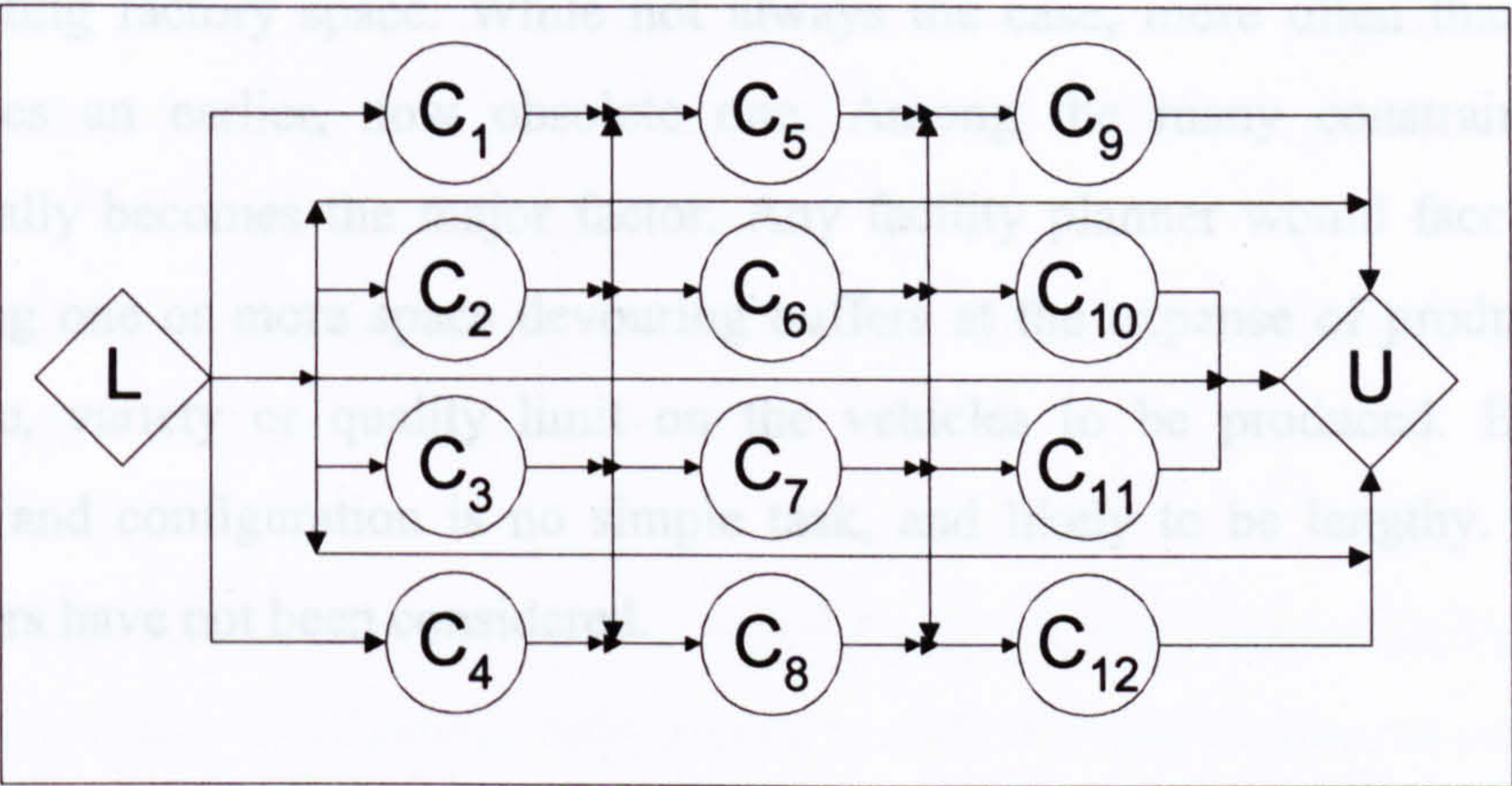


Fig 2.5 A hypothetical agile facility, where L is the loader (the component/subassembly source), C 1-12 are the cells and U is the unloader (framing station or sink). There may be more than one loader and unloader, depending on the required production rate and similarity between models. The cell number and layout is for illustration purposes only. Cells contain human and/or robot -guided processes, or may be buffers. Independent cell access and exit are essential.

In Fig. 2.5 each of those cells is able to send a subassembly to any other cell in the subsequent column, or ‘stage’. Thus a tooling pallet emerging from cell C2 may be sent to any cell from C5 to C12 or directly to any of the framing stations/unloaders. The mechanism for this is the following: The tooling containing the subassembly enters its first cell (usually, but not necessarily, one of C1 - C4), is processed, and waits until the controller gives it the go-ahead to exit and begin its transfer to another cell, selected according to the following criteria:

- The specific vehicle processing requirements,
- The availability of a cell which matches the above processing requirements,
- The location of the corresponding parts in the delivery system, and
- The necessary order of assembly (if applicable).

2.2.5 Buffers

Once a match is made, the controller moves the vehicle from the current cell (or buffer) to the newly selected cell. Depending on the controller priorities, this may not be the first tooling/subassembly in line (requiring processing), but the first one that satisfies all of the above variables. The presence of buffers is well known to enhance intercell traffic flow by allowing ‘ready’ subassemblies to be immediately available as soon as its next designated cell is available. However their presence is not always essential or even necessary, and in some circumstances, may not be beneficial. As an example, consider designing a production facility

inside an existing factory space. While not always the case, more often than not the new facility replaces an earlier, now obsolete one. Among the many constraints, the space limitation usually becomes the major factor. Any facility planner would face the choice of either including one or more space devouring buffers at the expense of production cells, or face a volume, variety or quality limit on the vehicles to be produced. Either way, its determination and configuration is no simple task, and likely to be lengthy. At present, in SIMAID buffers have not been considered.

2.26 Scheduling

Finally, mention must be made of the role scheduling plays in production. Extensive literature exists on this topic for linear facilities but very little for agile facilities. The main reason for this is that scheduling is usually done after a layout is known, but by definition, in an agile (constantly changing) facility, the layout also changes with time. So do the types and volume of vehicles required to be produced, typical of a make-to-order production system. In a 3-day delivery promise production philosophy, a shift's production requirements are not usually known until the day before, hence it is difficult to imagine any scheduling system that could cope with these demands unless it is a real-time one, incorporated into the system controller and continuously functioning to create last-minute and up-to-date schedules. This is the system that is best described by SIMAID.

2.3 Summary of the main assumptions used in this work

The main assumptions SIMAID considers are thus the following:

- All of the sub-assemblies are loaded onto a mobile tooling (either on a conveyor or an AGV) for processing and are carried through the process cells, as required, before reaching the unloader/framing station;
- Each sub-assembly process combination takes a variable amount of time to perform, hence there is no fixed cycle time;
- Strict process order preferences are taken into account;
- The tooling transport speed can be fixed at the start or made open to changes within limits by the program; it is initially set at 1 m/ second;
- Cell layout is designed with agile principles in mind and thus does not follow the conventional 'line' layout.

- Loading a set of parts onto the tooling takes a finite but not insignificant time;
- Anywhere between 1 and a maximum of 4 robots is allowed per cell to avoid excessive interference between them;
- Both inter-operation time lag and multi-robot hindrance factors are taken into account during the processing phases.

Finally, the inclusion of further options such as toolchangers, while desirable, has not been included. The advantage of allowing toolchangers in cells is in circumstances such as when a specific process, while essential, is little used and this brings the whole cell’s utilisation down. With a toolchanger the possibility of adding one or more processes to that cell, and thus increasing its utilisation rate, is both real and desirable, particularly when time, space or cost are strongly limiting factors.

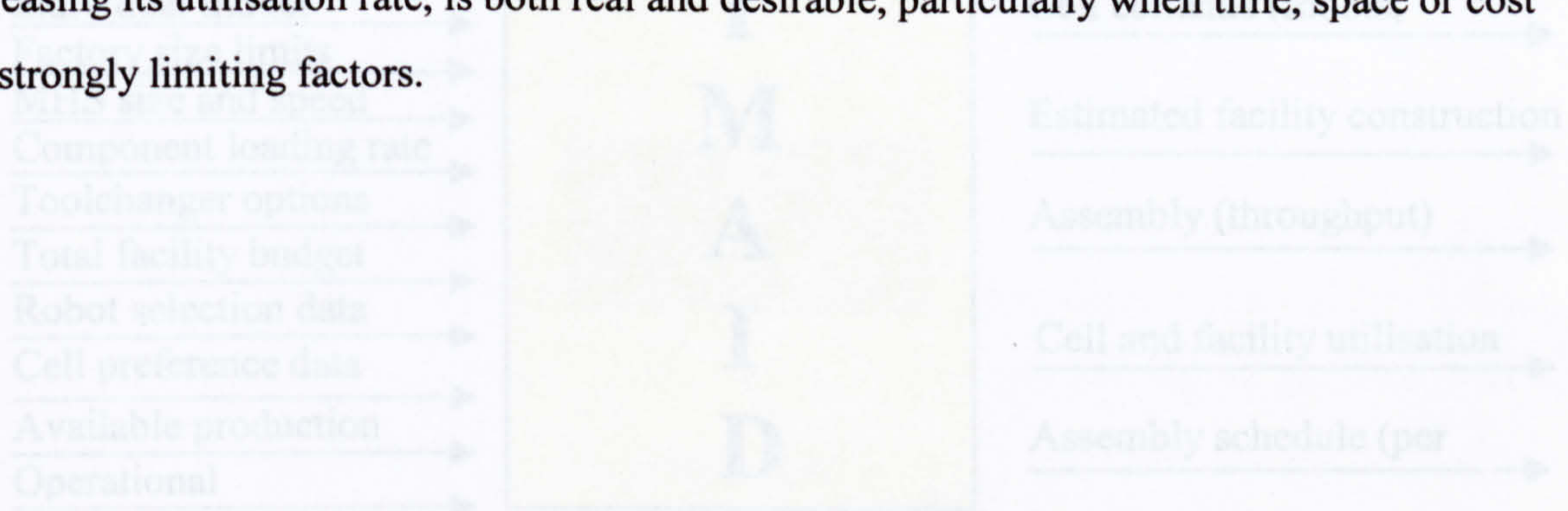


Fig 3.1 The SIMAID model. *The operational characteristics are number of shifts, multi-robot cell operation times, inter-process robot arm movement time, etc.

The program is a menu-driven GUI (graphical user interface), starting from the opening window, which is the root. The reason for this type of GUI (as opposed to a purely textual data entry format) is simply ease of use and understanding. An important aspect is that the user must be able to understand exactly what is occurring, and thus both the data entry format and its execution presentation should be graphical.

The algorithm functions in three stages: the first stage calculates the basic requirements for production and generates a first layout. The second stage schedules the subassemblies and the third stage enters an iterative evaluation, solution search and improvement loop. The end result is a recommendation for the best layout it can provide, given the objectives and constraints. At each stage there is the option for the user to go back and modify the entry data and re-run the sequence. It is important to note that this has been incorporated because the

Chapter 3 – Methodology

3.1 The model

The methodology is split into two chapters, Chapters 3 and 4. This chapter gives the general overview so that the ideas behind the program can be understood. Chapter 4 gives further details and the mathematical principles. The general model of SIMAID is shown below. On the left the input data SIMAID requires to function. On the right are the outputs, seen in both visual and data formats.

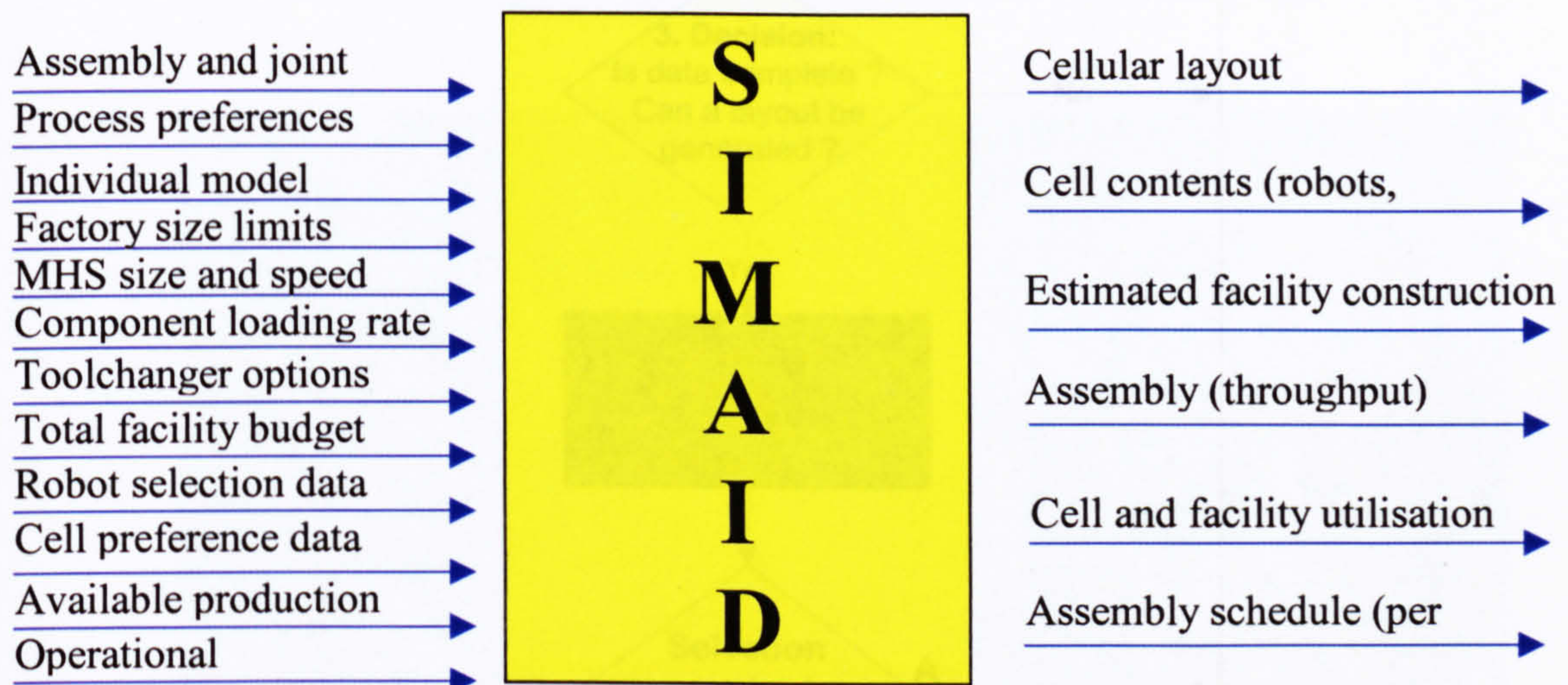


Fig 3.1 The overall model. *The operational characteristics are number of shifts, multi-robot cell operation times, inter-process robot arm movement time, etc.

The program is a menu-driven GUI (graphical user interface), starting from the opening window, which is the root. The reason for this type of GUI (as opposed to a purely textual data entry format) is simply ease of use and understanding. An important aspect is that the user must be able to understand exactly what is occurring, and thus both the data entry format and its execution presentation should be graphical.

The algorithm functions in three stages: the first stage calculates the basic requirements for production and generates a first layout. The second stage schedules the subassemblies and the third stage enters an iterative simulation, solution search and improvement loop. The end result is a recommendation for the best layout it can provide, given the objectives and constraints. At each stage there is the option for the user to go back and modify the entry data and re-run the sequence. It is important to note that this has been incorporated because the

finding of a feasible solution is not assured. This depends entirely on the objectives and constraints imposed by the user. Fig. 3.2 below displays the general program flow:

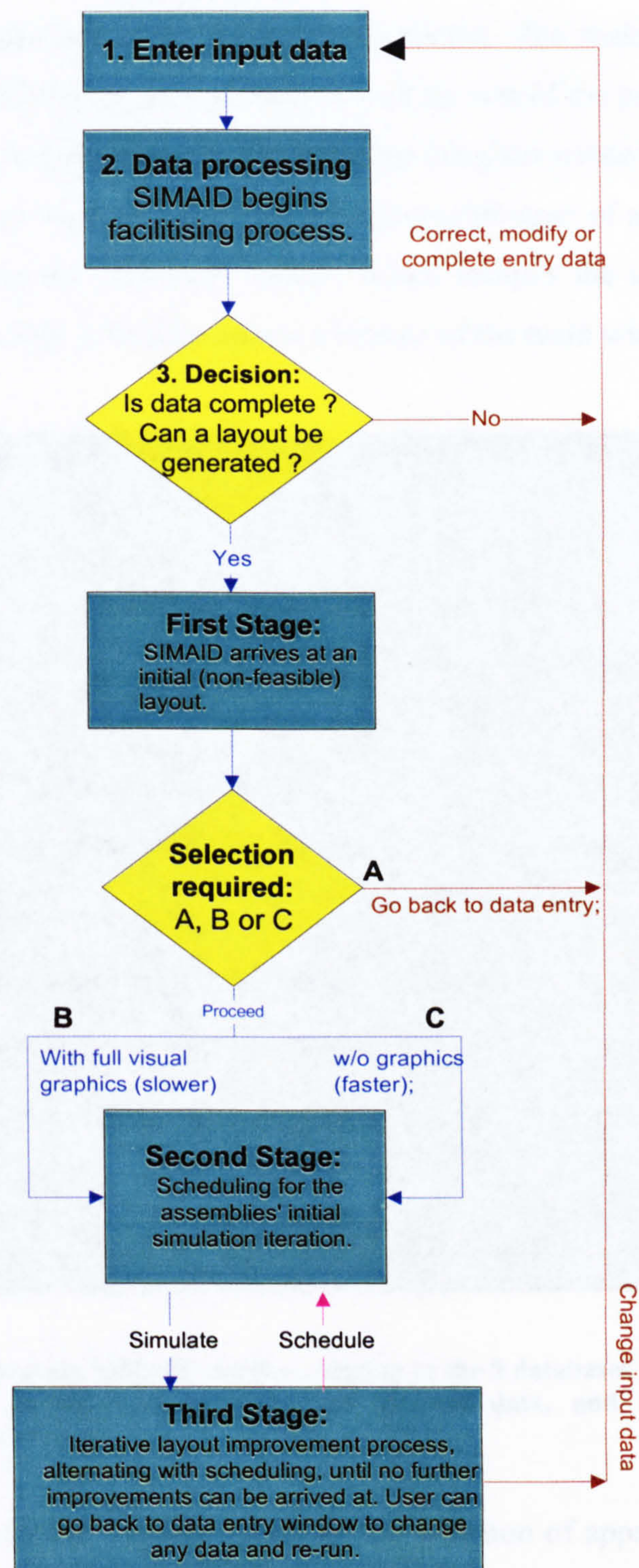


Fig 3.2 Top-level flowchart summarising SIMAID's overall functionality, showing the three consecutive stages.

3.2 The program

The initial layout is determined using assembly and production data through a menu-GUI, which displays 5 databases and other relevant information. The main GUI window is the opening window, which allows the user to interact with the rest of the program. Each database for the input of the above information is inside a further daughter window, called up by named buttons placed on the main window. Following the successful input of all of the required data, the program then displays the “Proceed” button, which enables the user to commence the layout formation process. Fig. 3.3 below shows a bitmap of the main window:

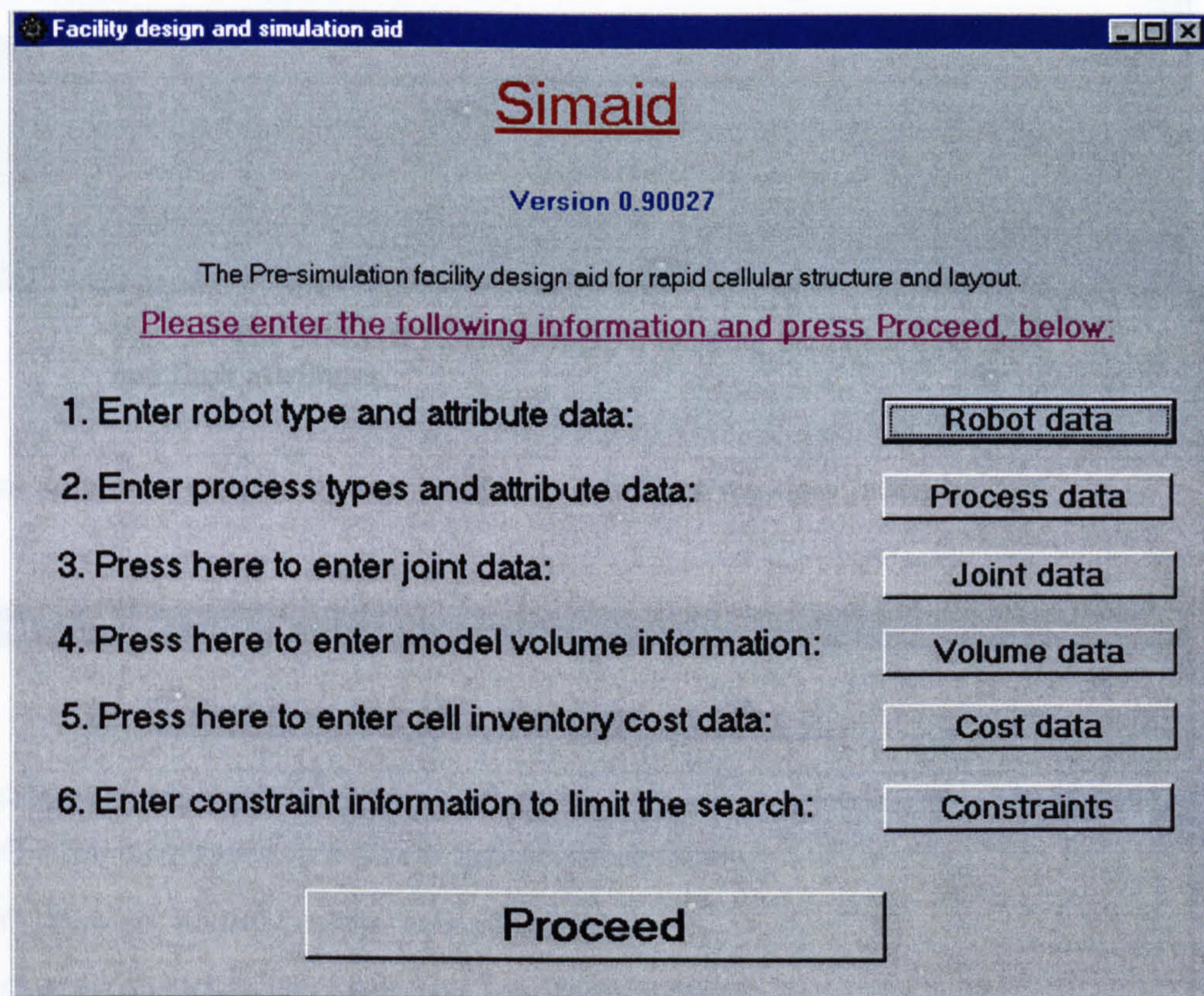


Fig. 3.3 The main SIMAID window, leading to the 5 databases for the input of the required part and process data, and the constraints window.

The first is a database of robot characteristics for the selection of appropriate robot(s) for the required processes. These characteristics have been designed to allow SIMAID to present the user with one of several options as to the robot selection method for the robots to be employed within the cell's facility. Examples are items such as the individual robot's payload,

repeatability, accuracy, minimum and maximum arm reach, maximum combined speed, etc. Fig. 3.4 below shows a bitmap of this window:

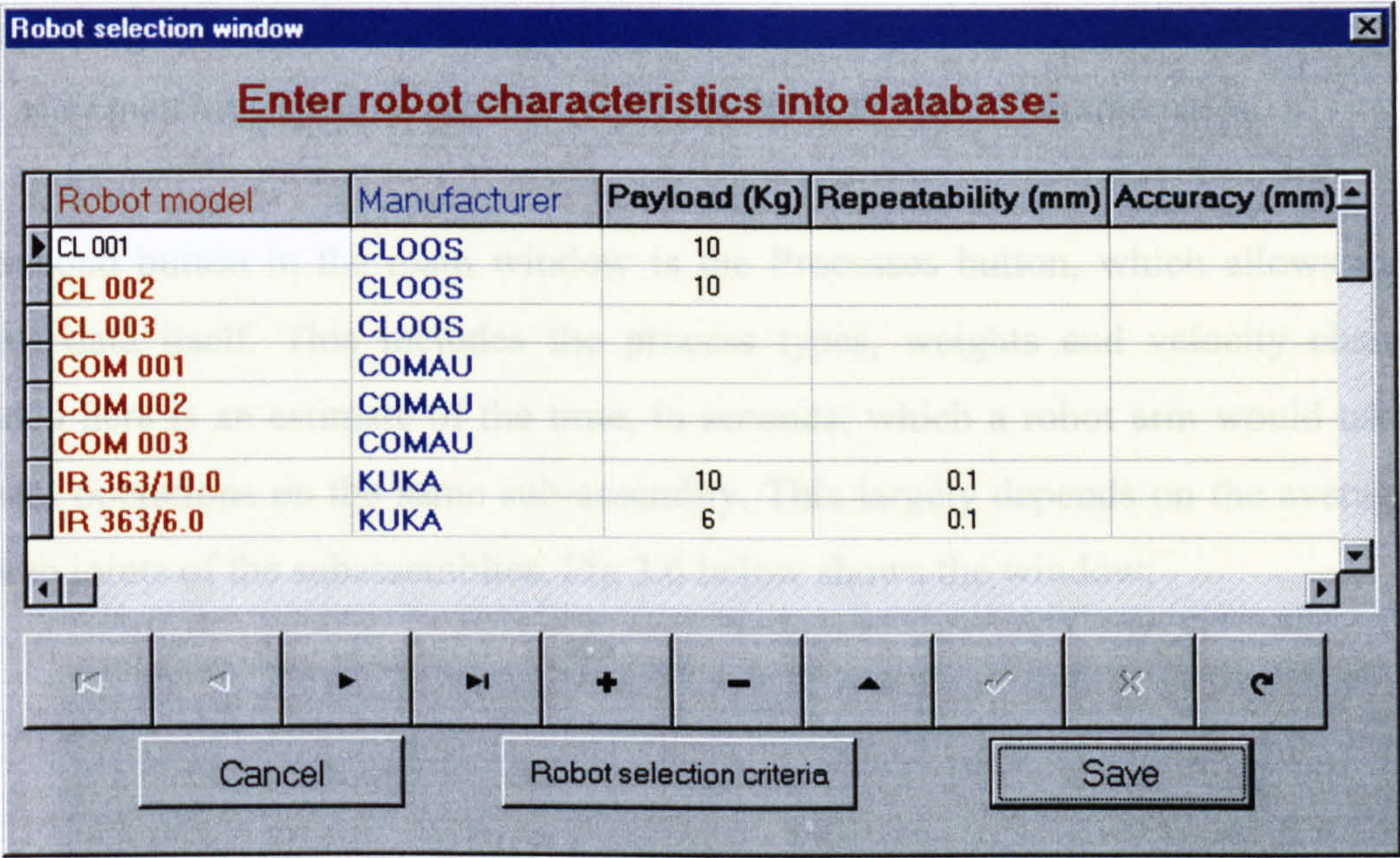


Fig. 3.4 The SIMAID Robots window, showing the robot databank and their attributes.

The robot selection criteria button brings up a further window, seen below:

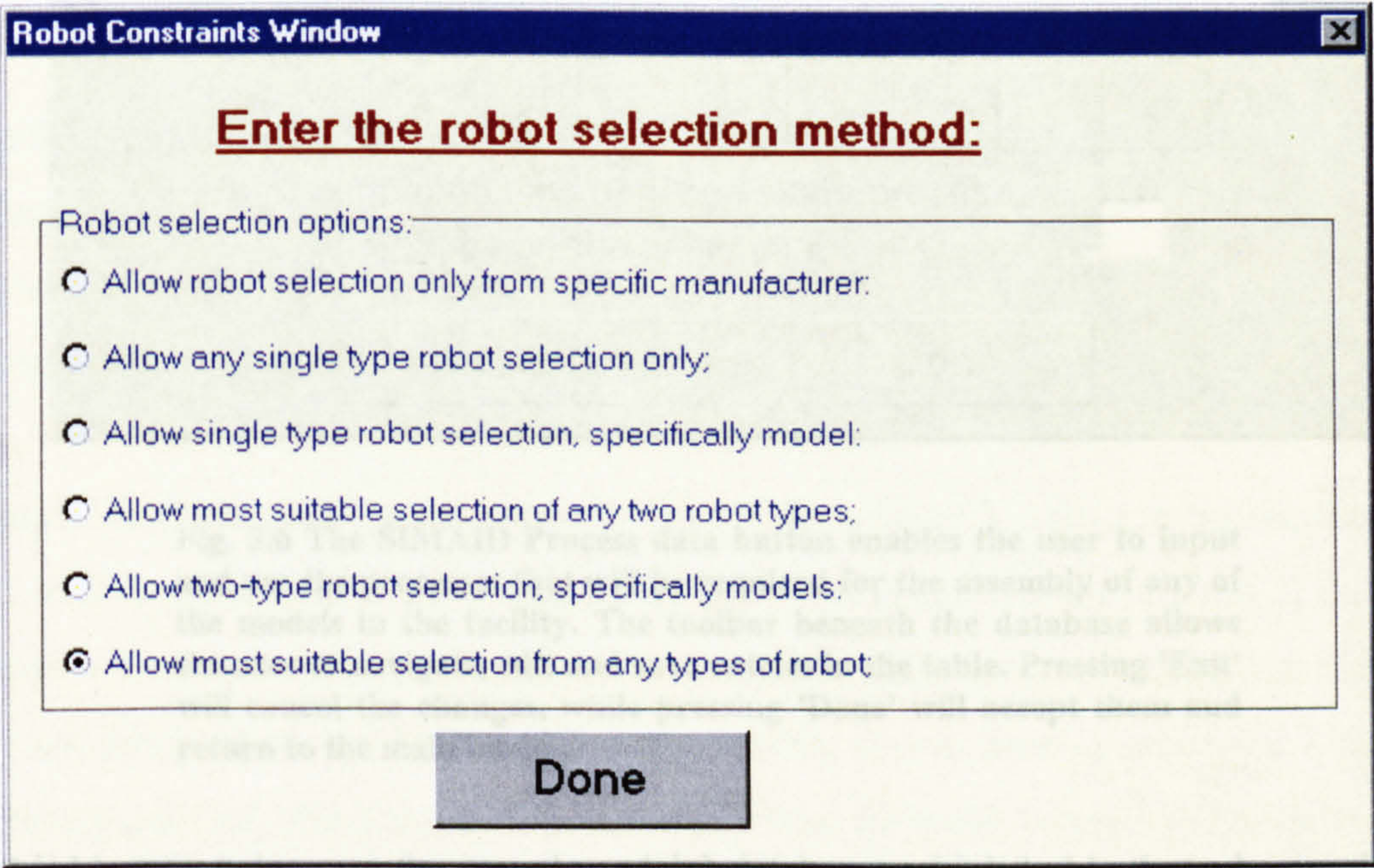


Fig. 3.5 The SIMAID robot selection criteria button enables the user to select the means by which SIMAID allocates robots to the cells.

As described, the user can select between any one or two models, any manufacturer, or allow SIMAID to allocate these robots according to requirements. Future developments envisage incorporating into the methodology a system whereby SIMAID selects the most adequate robot after matching specific process requirements with robot characteristics.

The second button in the main window is the Processes button, which allows input of the process data itself. This includes the process types, weights and velocity characteristics. Included here is an estimate of the time, in seconds, which a robot arm would take between adjacent operations on the same sub-assembly. This largely depends on the average distance between joints of the subassemblies. Fig 3.6 below shows the window:

Process selection window

Enter process type and attributes into database:

Process name	Tool weight (Kg)	Tool speed (Units/time)	Units
Spot welding - Type 1	3	0.3571428	Spots/Sec
Spot welding - Type 2	3	0.3571428	Spots/Sec
Spot welding - Type 3	3	0.3571428	Spots/Sec
Spot welding - Type 4	3	0.3571428	Spots/Sec
Spot welding - Type 5	3	0.3571428	Spots/Sec
Spot welding - Type 6	2	0.3571428	Spots/Sec
Bolting	5	0.2857142	Bolts/Sec
Adhesive application		0.2	Metres/Sec

Navigation buttons: [Previous], [First], [Next], [Last], [Add], [Delete], [Up], [Down], [Check], [Cancel], [Refresh]

Enter estimated time between same-process operations on the same joint :

Exit **Done**

Fig. 3.6 The SIMAID Process data button enables the user to input and use the processes that will be required for the assembly of any of the models in the facility. The toolbar beneath the database allows the user to navigate, edit and save entries in the table. Pressing 'Exit' will cancel the changes, while pressing 'Done' will accept them and return to the main menu.

The third button brings up the actual models' database, which holds the subassemblies' joint data. This contains specific joint process requirements, in terms of the processes described in

Database 2, such as type, order and quantity of each process. Fig 3.7 below shows the window:

Assembly joints window

Enter joint data:

Joint name	P. required	P. preferences	Process1	Process2	Process3	Process4
m1, stage6: joint 1	1,3,6,7	136	4		3	
m1, stage6: joint 2	2,3,6,7	236		4	3	
m1, stage6: joint 3	1,3,6,7	136	4		3	
m2, stage1: joint 1	1,2,3,4	0	3	3	3	
m2, stage1: joint 2	1,2,3,5	0	2	3	2	
m2, stage2: joint 1	1,2,3,4	0	2	3	4	
m2, stage2: joint 2	2,3,4,6	23		3	3	
m2, stage3: joint 1	1,3,5,7	13	3		4	
m2, stage3: joint 2	2,4,6,8	24		3		

Navigation buttons: ◀ ◻ ▶ ▶ ◻ + - ◻ ◻ ◻ ◻ ◻ ◻ ◻ ◻ ◻ ◻

Buttons: Exit Update database Save

Fig. 3.7 The SIMAID Process data button creates this window, which enables the user to input and use the processes that will be required for the assembly of any of the models that wish to be built within the facility.

Not all joints will require all processes, and usually only one or two at most, but as there will be numerous joints per subassembly, these may require all, most, or only some of the processes. The order in which these processes are needed is essential, and the database has a process preference section to cater for this.

Example: Referring to figure 3.7, above, the model 2, stage 3, joint 1 entry requires processes 1,3,5 and 7; the actual order of operation is first, process 1, then process 3, then either of processes 5 or 7, in any order. Process 5 may be an adhesive operation that may require a previous fastening or welding operation(s), or both, shown here as processes 1 and 3. These are then quantified, either in terms of time (e.g. time for weld, in seconds) or units (e.g. no. of rivets). Process 7 may be a deburring, cleaning, polishing, inspection or any other type of operation.

The fourth database is an extension of the third, as it is the result of a process calculation sequence. The visible result is a table with the joints listed with a summary of their process requirements next to them, in terms of type, order, start time and duration (in seconds) required. E.g.: m1, s1, j1 - p3: 0,9; p5: 10,6; etc., means the first joint (j1) of the first subassembly (s1), of the first model (m1) requires process 3 to begin at time zero and lasts for 9 seconds, immediately followed by process 5, whose duration is six seconds. There may be up to 4 processes per joint. This table is created so the user has a chance to analyse and/or modify any joint data before the program uses it in the subsequent stages.

The fourth button displays the Volume data window, where the individual model volumes are entered. At the top is the total volume of vehicles to be produced within one calendar year (to be defined later on). Beneath is the estimated total volume variation within the period, in this case, +10%. A negative value could also be entered. Finally, there are the individual model entry boxes. Fig 3.8 illustrates this:

The screenshot shows a window titled "Volume data" with a close button (X) in the top right corner. The window contains the following elements:

- a) Enter the total expected vehicle volume per year:
- b) Enter the forecasted volume fluctuation percentage:
- c) Enter the expected relative model volumes:

Model 1 <input type="text" value="100000"/>	Model 3 <input type="text" value="30000"/>
Model 2 <input type="text" value="50000"/>	Model 4 <input type="text" value="0"/>
- At the bottom, there is a button labeled "Check data".

Fig. 3.8. The 'Volume' button brings up the 'Volume data' window for entering the individual production volumes of the models that are to be built by the facility. Entry b) allows the user to provide some production volume flexibility, telling SIMAID to expect (in this example) a potential 10% higher production level in its layout. Once the data is inputted, the 'Check data' button checks its consistency and if correct will return the user to the main window.

The fifth database is the cost database, where every type of equipment that can be used in constructing the facility is costed. Provisions could be made for the non-linear relationship between quantity and price, but this hasn't been done due to the enormous variation in such estimates. Hence the input cost is consistently used, whether for a single robot or 100. Also included is a cost for space (area). Fig 3.9 below shows the window:

Costs data input

Enter cell and facility cost data:

Item to cost	1	10
MHS, Y- transfer point		
Other item		
P1 robot		25
P2 robot		25
P3 robot		25
P4 robot		25
P5 robot		25
P6 robot		25
P7 robot		25
P8 robot		25
Proc1: Adh.Injection		40
Proc2: Mig welding		20
Proc3: SPR		25
Proc4: Spot clinching		20
Proc5: Spot welding		25
Proc6: SDDS		20
Proc7: Laser welding		80
Proc8: Hammer job		10
Toolchanger, 2-process		10
Toolchanger, 3-process		12
Tooling		60

OK
Cancel

Navigation buttons: Previous, Next, First, Last, Add, Subtract, Up, Down, Check, X, Refresh

Fig. 3.9 The 'Costs' button brings up the costs database window for entering the individual and quantity costs for the equipment and tooling to be used in designing the facility.

The final button on the Main window is the constraints button, which brings up a window to allow the user to input the following data:

- Maximum facility cost;
- Maximum facility length;
- Maximum facility width;
- Maximum number of routing alternatives;

- Maximum number of robots per cell;
- Maximum number of shifts per 24 hours;
- Hours and minutes available per shift;
- Available days and weeks per year.

Constraints data input

Enter constraint data to limit search range:

Maximum facility cost (£): £: 3000000

Maximum facility Length (in metres): 50 metres

Maximum facility Width (in metres): 30 metres

Minimum number of routing alternatives: 0

Maximum number of robots per cell: 4

Maximum number of shifts per 24 hours: 3

There are 6 hours and 36 minutes in each shift.

There are 48 working weeks a year of 5 days each.

Cell data Done Cancel

Fig. 3.10 The 'Constraints' button brings up the constraints data input window for entering the constraining data to be used in designing the facility. The only non-functioning entry here is 'Maximum number of robots per cell', which has not been implemented. The 'Cell data' button, which must be pressed before returning to the main window, brings up the cell data entry window (see Fig. 3.11 below).

The Cell data button brings up a daughter window to define the type of limitations to be included in the design. The data to be entered is:

- Assembly size, length and width, in metres;
- Tooling size, as above;
- Average component loading time, in seconds;
- MHS width, in metres;
- MHS: opposite directions placed vertically or horizontally in parallel; and finally,
- Toolchanger use, selection from: None, single/cell, multiple and under-utilised cell only.

Cell data window

Enter the following data:

Material Handling System width (in metres):

Average component loading time (secs):

Min tooling transport system's speed(m/s):

Max tooling transport system's speed(m/s):

Assembly size - length (in metres)

Assembly size - width (in metres)

☒ **MHS : Directions placed in parallel**
☐ **MHS : Directions placed vertically**

Toolchanger information

☐ Disallow toolchanger use, 1 process/robot and/or 1 robot/process only;
☐ Allow 4-tool (2-robot, 2-process) toolchangers only;
☐ Allow multiple process toolchangers in underutilised cells only;
☒ Allow multiple process toolchangers in any cell (let me do the thinking);

☒ **OK**

Fig. 3.11 The 'Cell data' window for entering the cell data to be used in designing the facility.

3.3 Stage 1: Initial layout derivation

Having entered the above requested information, the user can then press the Proceed button and, if all the required data is present and complete, SIMAID begins the first stage of layout design. The first stage uses a 3-phase group technology cell formation approach, using the above information to create an initial layout. This first layout is invariably a non-feasible solution - namely, the stated production volume cannot be built within the stated shift time and with the current cell configuration. The process of facility modification to arrive at feasible solutions occurs in the second and third stages. A schematic outline of the first stage's sequence is given in fig 3.12 below:

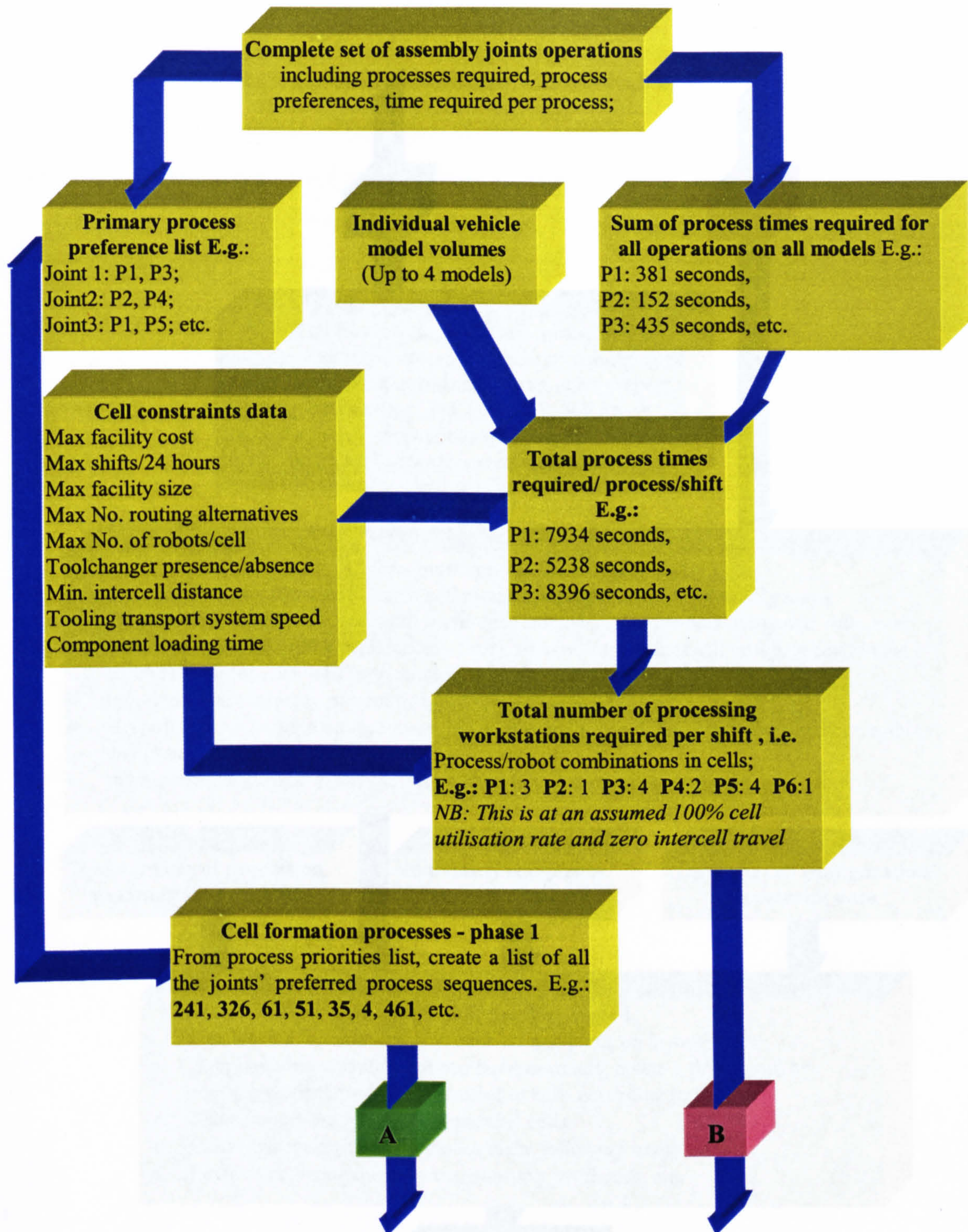


Fig. 3.12A Flowchart of the program based on the first stage of the SIMAID methodology, from raw process data to the generation of the process priority list.

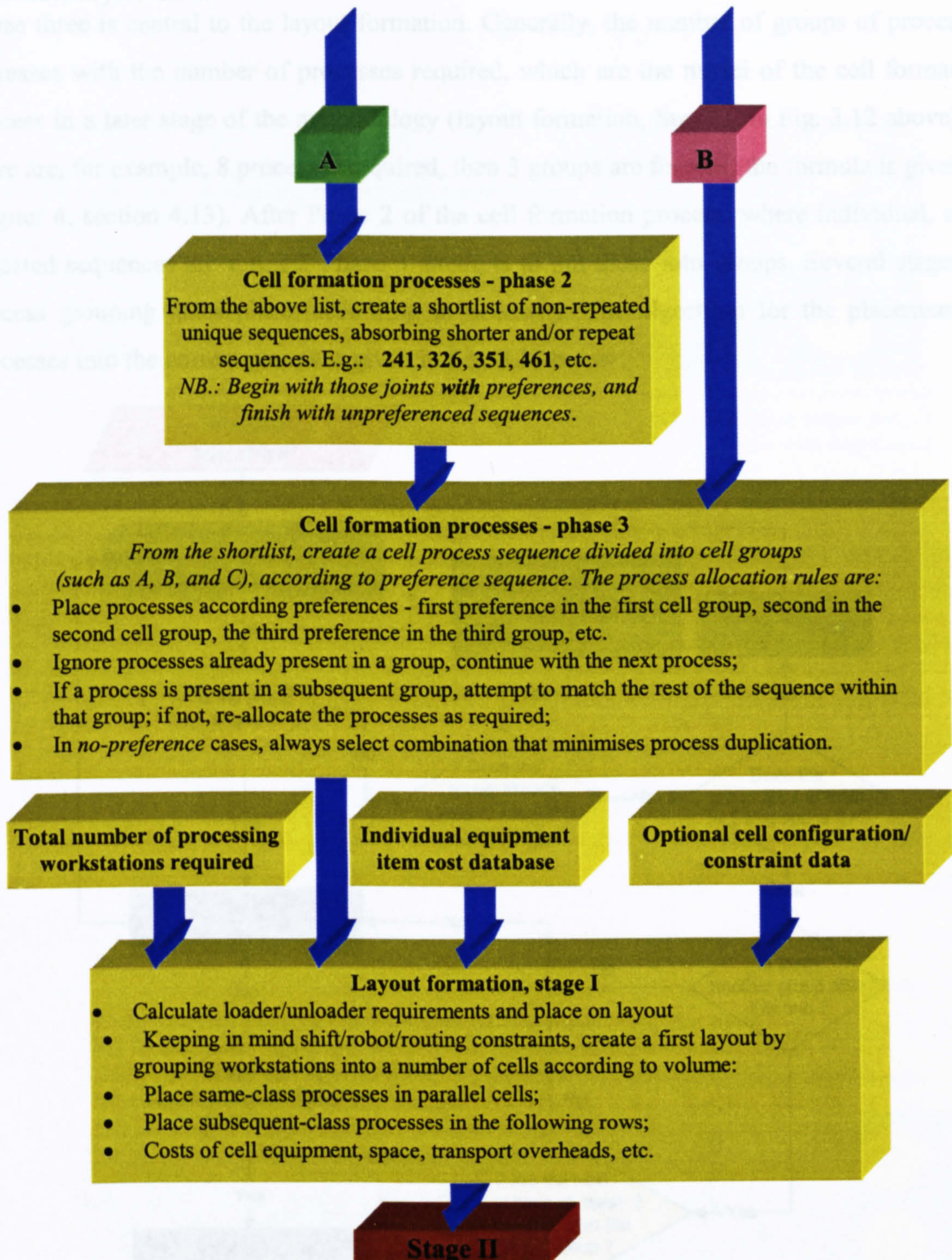


Fig. 3.12B Flowchart of the program based on the first stage of the SIMAID methodology, from process priority list to the generation of the first layout by a cell formation process. The full mathematical treatise is given in Chapter 4.

3.3.1 Layout formation

Phase three is central to the layout formation. Generally, the number of groups of processes increases with the number of processes required, which are the nuclei of the cell formation process in a later stage of the methodology (layout formation, Stage I, in Fig. 3.12 above). If there are, for example, 8 processes required, then 3 groups are formed (the formula is given in chapter 4, section 4.13). After Phase 2 of the cell formation process, where individual, non-repeated sequences are formed, Phase 3 attempts to put these into groups. Several stages of process grouping takes place, according to necessity. The algorithm for the placement of processes into the correct groups is given in Fig. 3.13 below:

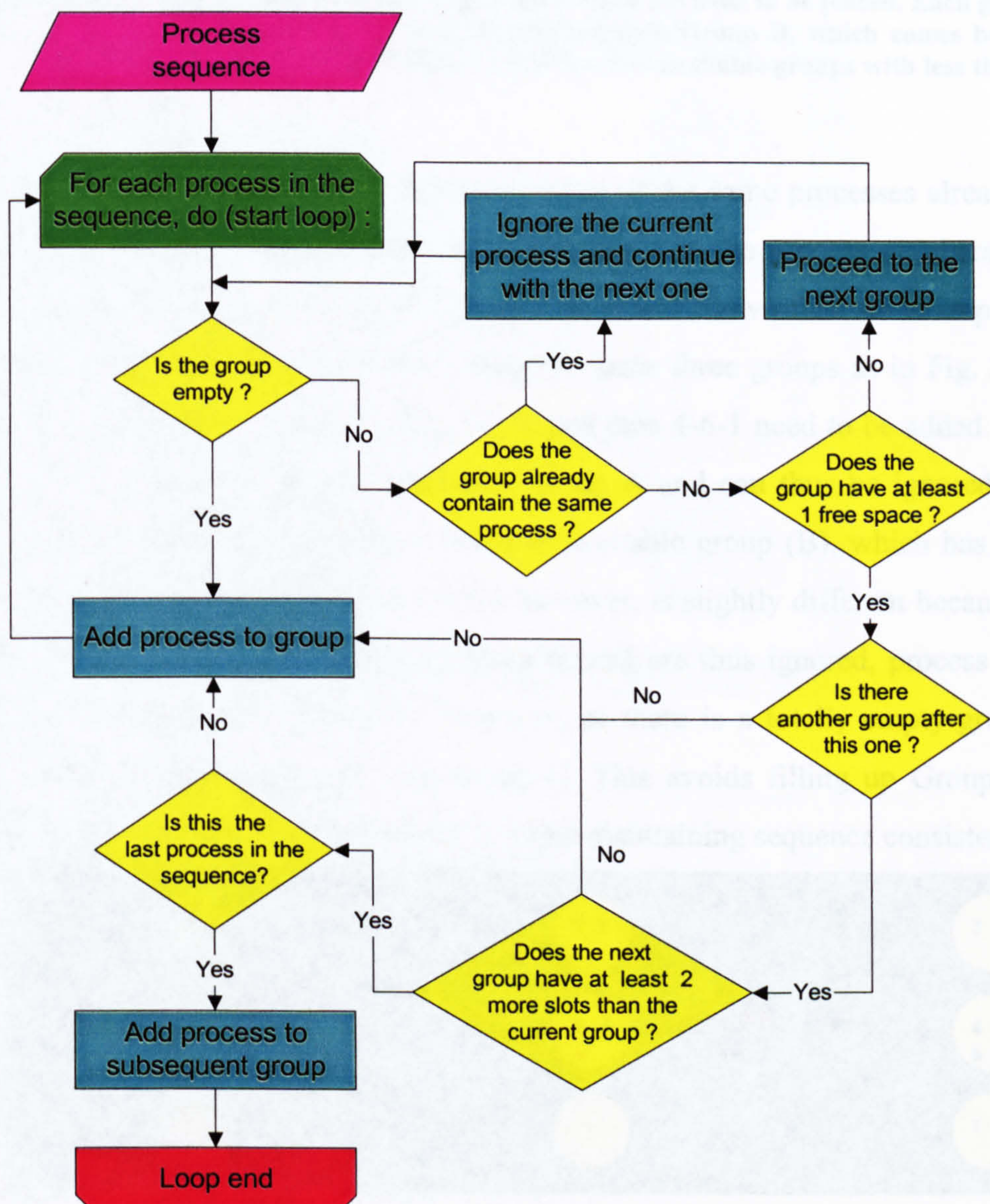


Fig 3.13. The process-placing algorithm in phase three of stage 1.

The first of these is the simple one of putting all the Phase 2 process sequences into the available groups. Fig. 3.14 below describes the first stage with an example:

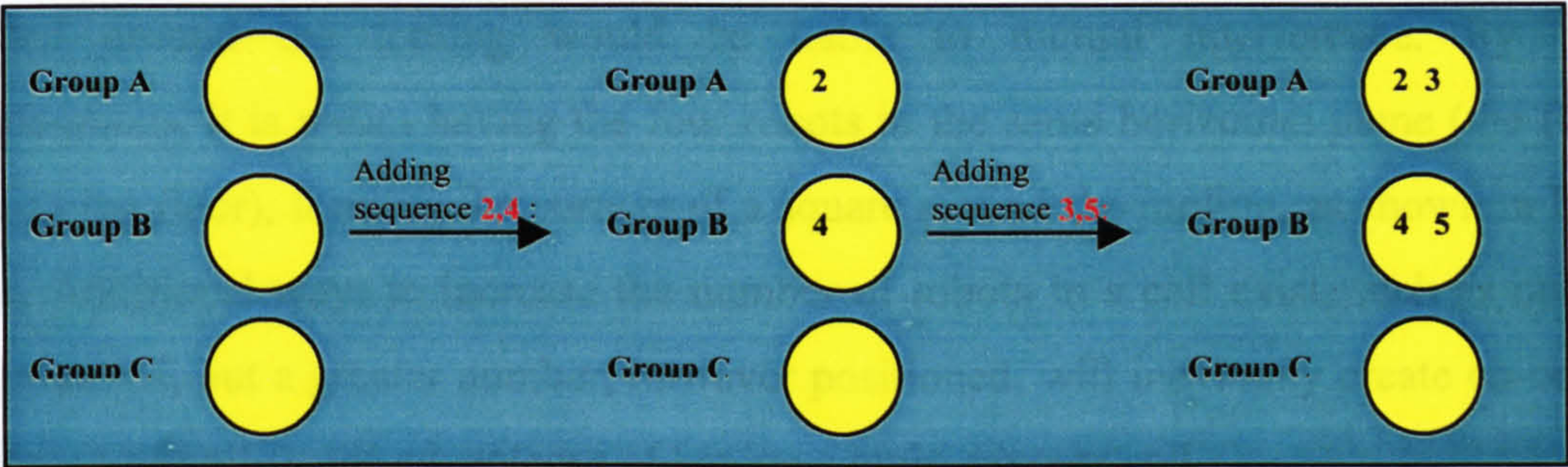


Fig 3.14. In this example, the two sequences 2-4 and 3-5 need to be placed. Each group has a maximum of 4 places. Group A comes before Group B, which comes before Group C. The algorithm places them in the first two available groups with less than 4 filled places.

In the case where the next sequence either has some of the same processes already installed into the groups, or where there is substantial imbalance in the process numbers within the groups, the algorithm places the sequence according to workflow rules. An example of how it works is given below in fig. 3.15 below, using the same three groups as in Fig. 3.14 above. Two further subassembly sequences, first 3-2-6, and then 4-6-1 need to be added. In the case of 3-2-6, both 3 and 2 are already present in group A and can thus be ignored; process 6 (shown in red) is placed in the next consecutive available group (B), which has two vacant slots. Placing the processes of sequence 4-6-1 however, is slightly different because although processes 4 and 6 are already present in group B, and are thus ignored, process 1 could be placed in either Group B or Group C. However, as there is a totally empty group next to Group B, SIMAID places process 1 in Group C. This avoids filling up Group B for any subsequent sequences that may need to use it, while maintaining sequence consistency.

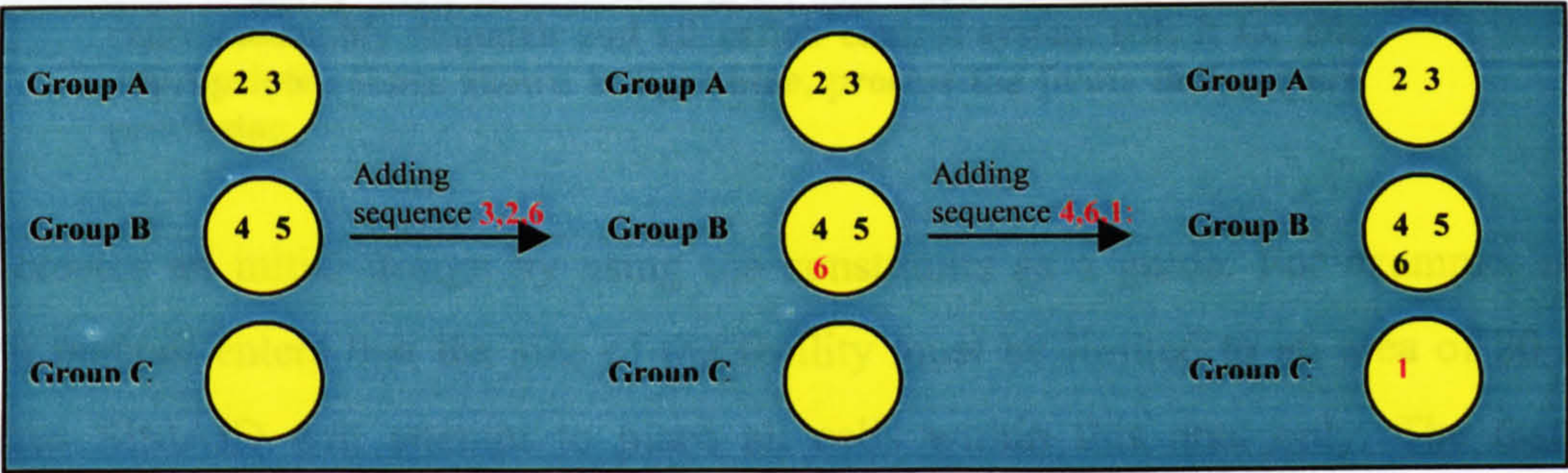


Fig 3.15. Incremental adding of processes with different requirements.

3.3.2 The cell

It is worth remembering, from the assumptions in Chapter 2, that we decided to use a maximum of 4 robots/processes per cell, as in 'normal' configurations any more of them crowded around the tooling would be liable to mutual interference. By 'normal' considerations, it is meant having the four robots in the same horizontal plane (the floor or a stand on the floor), forming the corners of a square around the tooling, as shown in Fig. 3.16 below. Additional ways to increase the number of robots in a cell exist, such as inverted or wall mounted, but a greater number, however positioned, will inevitably create co-ordination problems and nullify any advantages of having a multi-process cell.

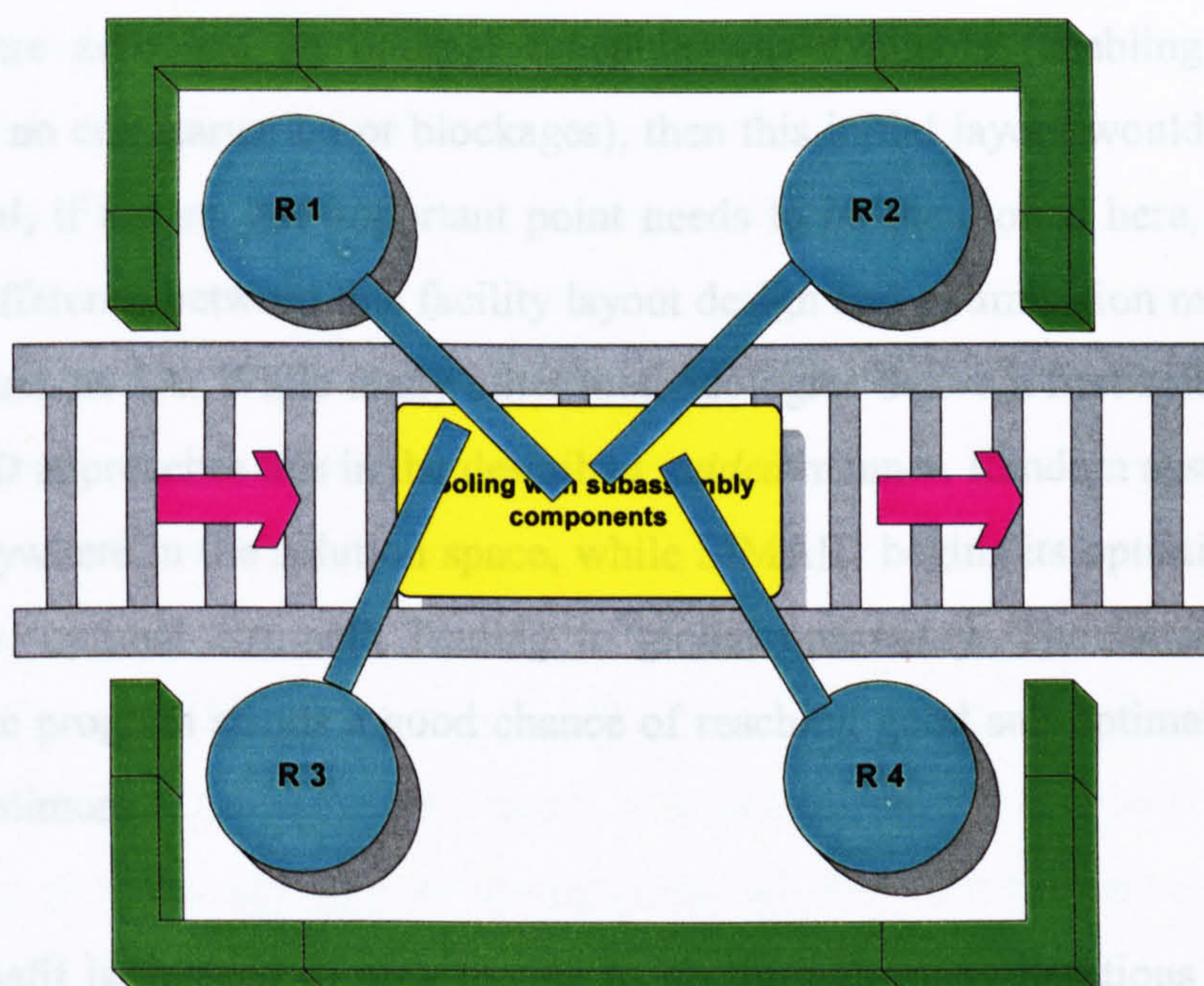


Fig. 3.16 The layout of a hypothetical cell with the maximum number of robots in 'normal' positions, R1 –R4. The component carrying tooling, in yellow, moves along the MHS, in grey, and stops within this cell if the cell/subassembly despatch and allocation control system tells it to. The appropriate robots, shown in light blue, process the joints that require processing.

SIMAID creates an initial design by using the constraints as a guide. For example, if the production planner enters that the size of the facility must be limited to an area of 20 x 30 metres, then SIMAID will attempt to place all cells within this area only. The distance between the cells is determined by the width of the MHS, and the number of robots/processes in each cell is determined by the amount entered, subject to a current limit of four (for demonstration purposes only).

3.3.3 The initial layout

From its creation, we know this layout is infeasible, because in the process of its creation there has been no consideration of:

- Inter-cell travel times,
- Cell availability patterns, resulting in possible cell blockage/starvation, and
- Entry/exit points for the tooling in the facility, namely the number and positions of the loaders and unloaders/framing stations.

By the other side of the coin it could be deduced that if, using an extreme case, intercellular travel time were zero and an optimal schedule was available (enabling 100% process utilisation, and no cell starvation or blockages), then this initial layout would be feasible and close to optimal, if not so. An important point needs to be mentioned here, underlining the fundamental difference between this facility layout design and optimisation methodology, and many others, such as SA. While many other methodologies derive a first solution by random means, SIMAID approaches this in the described *guided* manner. Random systems can initiate their search anywhere in the solution space, while SIMAID begins its optimisation iterations roughly in the ‘optimal channel’, leading to global optimality. The latter may never be reached, but the program stands a good chance of reaching good sub-optimal solutions close to the global optimum.

The major benefit is, instead of now having to go through many iterations to approach the optimal layout, SIMAID has already reached a design phase where optimality may be only a few iterations away, depending on the size of the problem. To summarise the reasons that this is so, in the first stage SIMAID forms a new layout based on:

- The actual process requirements of the subassemblies,
- The individual model production volume required,
- The constraints and limitations imposed upon it.

Due to the scheduling and travel time issues, the first layout is always unfeasible. Thus, the next stage must deal with one of the two problems, namely, scheduling.

3.4 Stage II: Scheduling

This stage is a preparatory one for the final stage, which relies on the simulation of the required shift's production for the layout generated. However, as the models and subassemblies will be in the order they have been put into the database, the issue of scheduling has not been addressed yet. Scheduling is important as a poor production schedule inevitably creates inefficiencies due to cell blockage and starvation, leading to a longer makespan than necessary.

3.4.1 The scheduling problem

The question of optimality for the purpose of scheduling is fuzzy in that it is extremely difficult to verify an optimal schedule for large numbers of subassemblies. The number of possibilities rises exponentially with each increase – $3! = 6$, $4! = 24$, $5! = 120$, $6! = 720$, etc., and even a small sequence of 10 subassemblies could potentially be ordered in 3,628,800 different ways. Given sufficient computer power (the 'brute force' approach - perhaps with several processors in parallel) and time, excellent, perhaps optimal solutions could be derived, but the need to operate within conventional PC limitations makes this clearly unmanageable. As an example, the scheduling of 20 subassemblies, a perfectly reasonable amount, would give us 2.43×10^{18} possibilities. Assuming a PC can solve a solution each nanosecond, which would mean one *billion* solutions per second. It would still take ~ 800 years to make sure to find the optimal solution, having gone through them all. For 21 subassemblies, this is 16,800 years [Lawler, 1975].

3.4.2 The SIMAID scheduling solution

SIMAID approaches this problem from another angle. It is clearly the exponential aspect of the problem that needs to be overcome; a large number of subassemblies may potentially need to be scheduled, but they are not actually all required at the same time. The number of units that affect, and are affected by, the facility layout design is actually much smaller. When SIMAID reaches the end of phase I, having designed its initial (infeasible) layout, we may have anywhere between 1 cell and 10, though typically, for a 3-model 100,000 unit production facility we may get around 3 or 4. Ideally, we would like to schedule every unit that needs to go through the facility, but, as explained before, this is impractical. As the minimum number should be the number of units that affect the facility, if this has, say, 4 cells that can only hold

one unit each, it makes sense to schedule at least those four. However, this is not sufficient, for as soon as one is done another is waiting to take its place from the queue. Thus, an additional number corresponding to the number of units that are liable to be passed through it in the ‘immediate’ future is required. In terms of an agile facility, we need to assume that:

- Each cell is independent, both in terms of accessibility on the part of the unit, and in its operation;
- A control system exists in place that enables each subassembly to be sent to a cell when this both matches its processing requirements *and* is available (free of units); and
- The scheduling must take into account the fact that in most cases there is the equivalent of a ‘framing station’, or unit ‘joining’ cell, for each model (or combination of models);

What this means is that scheduling must be done not just based on units getting through the facility but also consistent with the aim of putting them all together at the end. If the facility is bufferless, this has an even greater impact, as there is no other place to store those units already processed except in the cells themselves, and this is counterproductive as it blocks the cell for operation for subsequent units. To conclude, it makes sense to schedule a ‘batch’ of units at a time, but only all from the same model until all the units of that model are included before considering others.

Returning to the number of ‘additional’ units, we need to assume a worst-case scenario and plan for it. Assuming every cell is busy, we have a facility of n cells and u units in it. But we do not know at any point when any of the cells will finish its operation on the subassembly, as the processing times are varied. It may be that the units are freed uniformly or all at once, or anywhere in between. Assuming the worst case, that they are released all at once, and are making for the framing station, we can see that there are n empty cells now, and we need n units to fill them with. Hence, at any point in time, we will always need at least $2n$ units scheduled and ready to go. Fig. 3.17 illustrates this point:

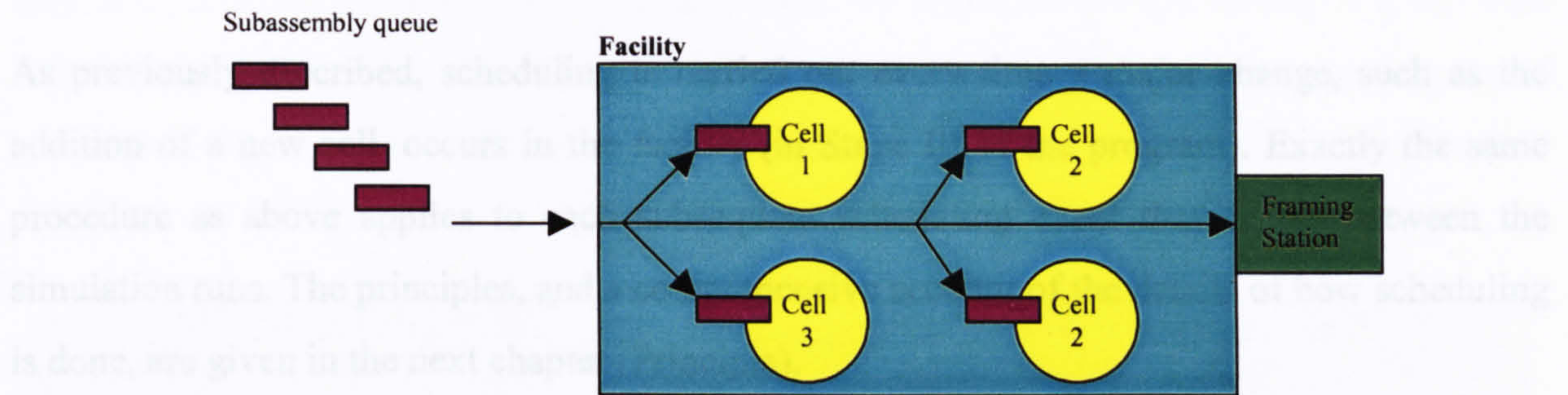


Fig. 3.17 Diagram showing the 'twice-the-cell-number' scheduling concept.

Thus, in this example, we may initially need to schedule around 8 subassemblies, but as SIMAID progresses and cells are added this may go up to 40 subassemblies (20 cells, SIMAID's maximum limit); the total number of possible combinations is $8.163e+^{47}$, a clearly unmanageable figure. Instead of scheduling a massive amount in one go, this work attempts to continually schedule smaller amounts in 'real time' - i.e. as SIMAID is working on an iteration. Early indications of this approach, though implemented as a separate program (i.e. not within SIMAID) and still under development, are encouraging. Fig 3.18 below gives the overview of the scheduling method employed.

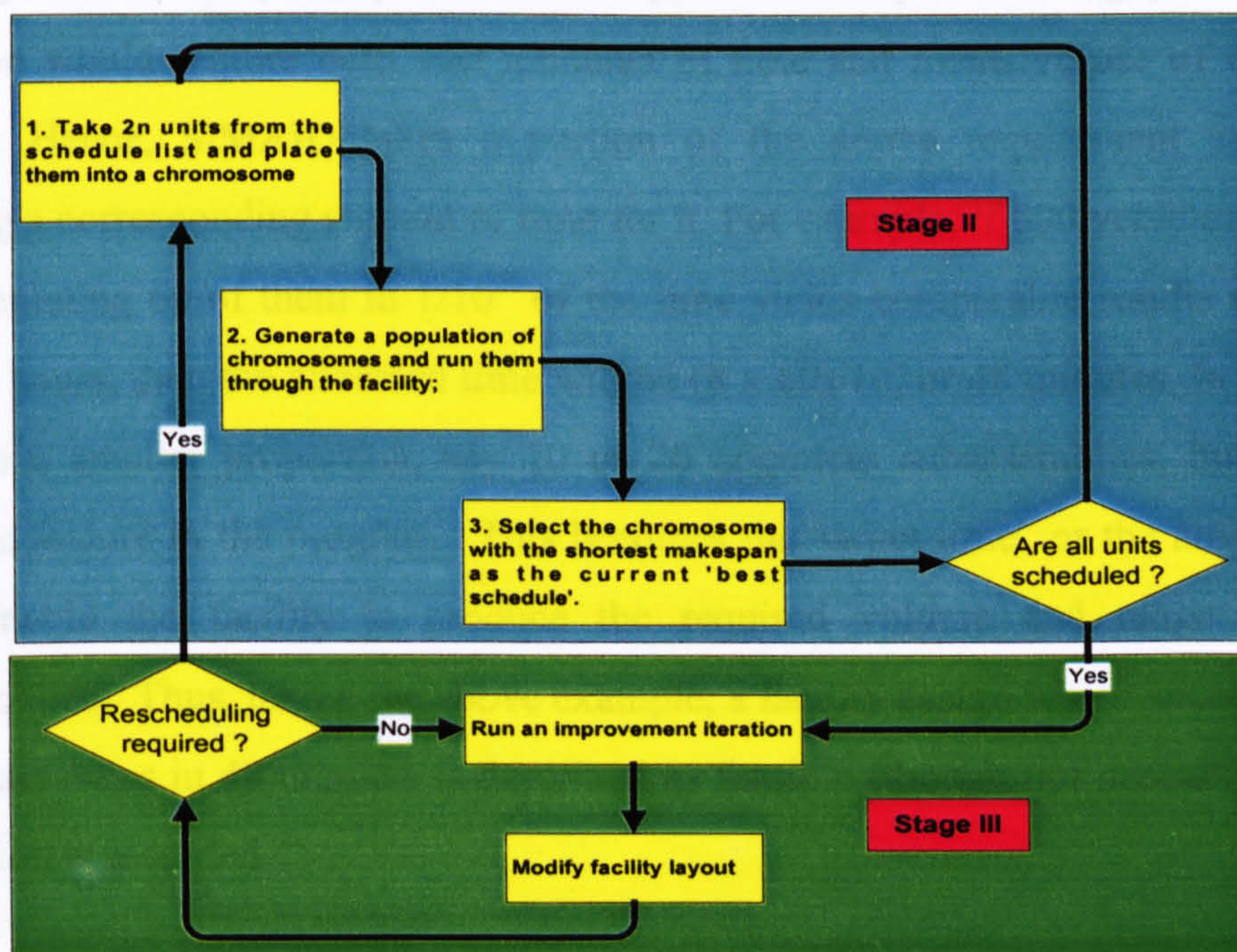


Fig 3.18 Flowchart showing a general overview of Stage II – the scheduling of the subassemblies prior to the improvement simulation runs of Stage III. As can be seen, this is a repeat process that occurs every time there is a major change to the facility design in stage III, such as the addition of a new cell.

As previously described, scheduling is carried out every time a major change, such as the addition of a new cell, occurs in the facility (in Stage III of the program). Exactly the same procedure as above applies to each subsequent scheduling event that occurs between the simulation runs. The principles, and a comprehensive account of the details of how scheduling is done, are given in the next chapter (Principia).

3.5 Stage III: Layout optimisation by iterative simulation of cell formation

This is the final stage where iterative layout improvement is carried out to a specific set of objectives and constraints. We now have an initial layout and a set of scheduled subassemblies to be processed per shift, according to their relative models' volume. As this initial layout is infeasible, we need to modify it in steps, approaching more feasible solutions.

3.5.1 The quantity of subassemblies to be simulated

Stage 3 simulates the processing of the subassemblies through the facility. However, if, for example, 3 models of 100,000 vehicles each need to be built, each has 12 subassemblies, and on average there are 10 joints per subassembly, the total required throughput is very large. The simulation would require both vast amounts of time and intensive use of CPU, and is not practical. Hence, SIMAID takes a portion of the above requirement and simulates it, allocating a corresponding amount of time for it. For example, if 600 vehicles are required per shift, simulating 60 of them in $1/10^{\text{th}}$ of the time yields comparable results much faster; if a shift is 8 hours, then the allocated time will be $(8 \times 60)/10$, or 48 minutes. In reality, SIMAID may take a smaller proportion, say 10 or 20 complete subassemblies, but this is a user-selectable feature in the program. This becomes the 'target time', or the amount of time that would enable the facility to produce the required volume and range of vehicles (or subassemblies). Thus, taking the above example, a facility design which successfully produces 60 subassemblies in 48 minutes is described as feasible (though not necessarily optimal) and the solution is recorded.

The above method would work very well if the 'slice' of production measured were taken from around the middle of the production shift. However, if the measure is taken from the beginning or towards the end, then an incorrect measurement may result from under-utilised

cells and processes. Hence, there must be a 'warm-up' period where the facility is filled with WIP before and after the 'slice' is measured. SIMAID does this by increasing the amount of production. Fig. 3.19 below illustrated the concept.

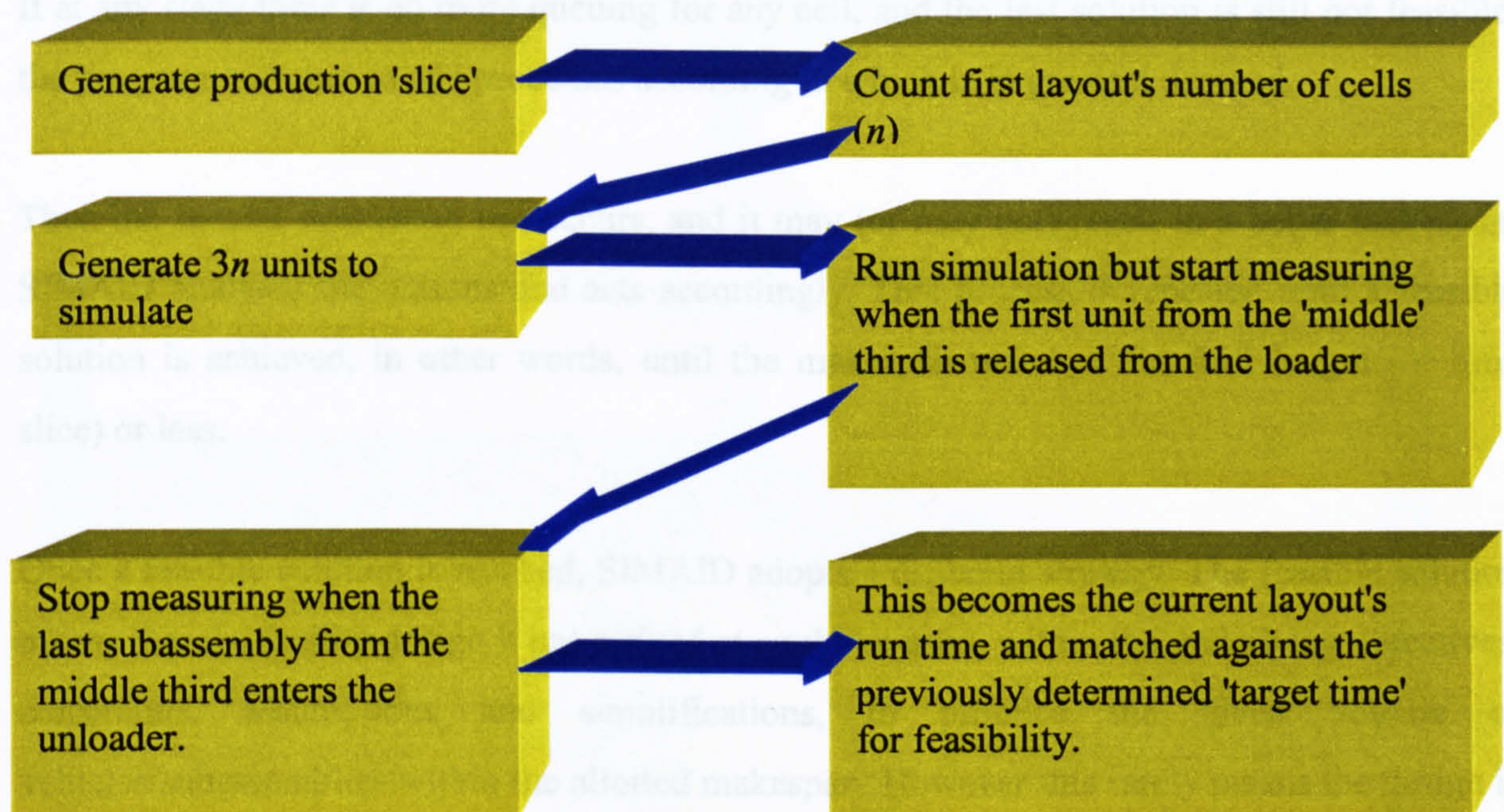


Fig 3.19 Flowchart showing a general overview of the warm-up period.

Since the initial generated solution is infeasible, the actual amount of time for the selected 'slice' of subassemblies to be processed through the current facility is likely to be higher than its target time. Experience shows that this can be anywhere between 150% - 300% of the required time. In the simulation run, all the subassemblies selected are pulled through the cells and processed as required, until the last one reaches the unloader. The makespan is then measured, and since it will invariably be greater than the target time, the first improvement iteration begins.

3.5.2 Optimisation

SIMAID analyses the performance of the first simulation run, identifies the biggest problem, and attempts, with a single change, to correct this. Usually, the initial situation is described as one of insufficient specific process presence for the job quantity involved, and the program's response typically reflects this in the addition of the process most in demand within a cell. Its

measurement is carried out by first, measuring which cell has the biggest blocking effect on the subassemblies queuing up behind it. If there's more than one process present in the cell, the busiest process within it is selected. This process is duplicated and placed either into the same cell, into a neighbouring cell (within the same stage), or placed alone in a cell of its own. If at any stage there is no more queuing for any cell, and the last solution is still not feasible, the program switches to add processes according to which is in greatest demand.

Then the second simulation run occurs, and it may (or may not) result in a better makespan. SIMAID analyses the reasons and acts accordingly. This process is repeated until a feasible solution is achieved, in other words, until the makespan matches the shift length (or time slice) or less.

Once a feasible solution is reached, SIMAID adopts a different strategy. The feasible solution means that the facility design it has arrived at could be used, within the underlying objectives, constraints, assumptions and simplifications, to produce the given volume of vehicles/subassemblies within the allotted makespan. However this rarely means the design is optimal, or even anywhere near it. Therefore a different search engine modifies this facility design by removing any excess equipment (such as robots, processes, cells, etc.), while attempting to maintain feasibility. 'Excess', is described as equipment not essential for the production of the stated volume. SIMAID will iterate through different configurations until it cannot remove any other piece of equipment without affecting the design's feasibility, primarily unutilised (or under-utilised) resources. Details of the actual mechanism are given in chapter 4.

A point worth noting is that while attempting to remove unnecessary equipment in an iteration, the next one may become infeasible again. Should this happen, SIMAID returns to the original strategy of adding processes until a feasible solution is reached. Since the 'current' layout will be substantially different to the first feasible one, different processes may be added. Hence, the facility may evolve in a different direction as well. Fig 3.20 below gives the general overview of this third stage.

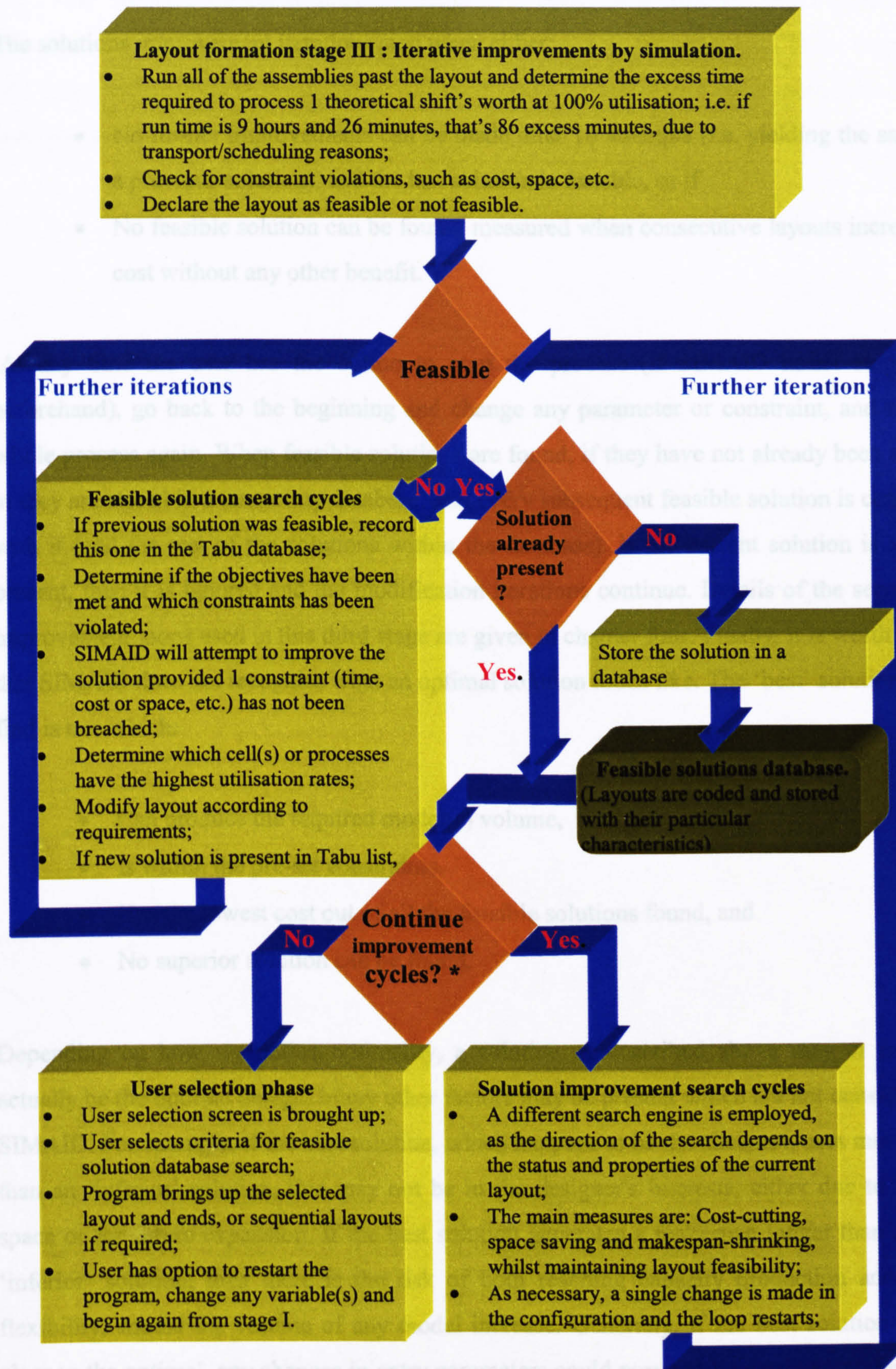


Fig 3.20. The final stage leading to the facility layout generation. *The cycle ends when there are no possible improvement, and the best solution is a feasible one.

The solutions improvement iterations stop when either:

- No further improvements can be made after 10 attempts (i.e. yielding the same or a previous solution), and the last solution is feasible, or if
- No feasible solution can be found, measured when consecutive layouts increase in cost without any other benefit.

At any time the user has the option to stop the process (if SIMAID hasn't stopped it beforehand), go back to the beginning and change any parameter or constraint, and run the whole process again. When feasible solutions are found, if they have not already been arrived at they are stored in a temporary database, and every subsequent feasible solution is compared with it (and the rest of the solutions within the database). If the current solution is already present, then it is ignored and the modification iterations continue. Details of the search and improvement loops used in this third stage are given in chapter four. Finally, it is worth noting that SIMAID does not recognise what an optimal solution looks like. The 'best' solution it can find is one which:

- Can produce the required model(s) volume,
- Is within the pre-set constraints,
- Has the lowest cost out of all the feasible solutions found, and
- No superior solution can be found.

Depending on how we define optimality, a solution as described above may or may not actually be the optimal design. Many other factors may be present which are not considered in SIMAID. For example, if the best solution, which happens to be the cheapest, uses more space than an 'inferior' solution, this may not be in the designer's interests, either due to lack of space or for future expansion. If the best solution generates a makespan longer than another 'inferior' solution, then there is the risk of both reaching capacity production and losing flexibility, should the volume of any model increase. Generally, if the best solution is very close to the optimal, any changes in entry parameters could prove to be detrimental. For all of these reasons, SIMAID has only been designed as a rapid facility design prototyping tool, and

any 'best' solution should then be exported to a commercial discrete event simulator to determine its suitability and modify to the designer's specific requirements. These may be an odd shape for available space, a different type of loader or component input mechanism, pallet transport system, etc.

Chapter 4 - Principia

This chapter describes the principles behind each stage in detail.

Definitions:

A *process* is defined as either an operation on a joint of a subassembly of a specific duration, or as a physical entity, such as a weld gun.

A *group* is defined as a collection of such processes placed together in a logical space, for the purpose of calculations.

A *workstation* is defined as a physical robot + process combination; this could be a spot weld or and adhesive application gun mounted on a robot arm, a multi-welding machine, etc. However, it does not necessarily need be a robotic process; a manual station is also applicable, such as a man with a filler gun.

A *cell* is defined as a physical space where one or more such workstations are present. It could also be a part transfer centre or a subassembly marriage cell.

4.1 Mathematical formulation of Stage I

4.1.1 Phase 1: Finding the workstation requirements.

Throughout this section, $\lceil n \rceil$ signifies rounding n to the nearest higher integer. The first action is to find the set of joint operations. Let

$$M := \{\text{models } m\}$$

$$S_m := \{\text{subassemblies } s \text{ in model } m\}$$

$$J_s := \{\text{joints } j \text{ in subassembly } s\}$$

Then the total time requirement, T_p , of process p is:

$$T_p = \sum_{m \in M} \left[\left(\sum_{s \in S_m} \sum_{j \in J_s} Q_{pj} \right) \cdot V_m \right] \quad \dots \quad (1)$$

Where:

Q_{pj} is the quantity value¹ of process p required for joint j , and

¹ Quantity value – this is an amount assigned to that process. For some processes it may be an integer (e.g. the number of rivets), a rate (amount of time of heating) or a volume (amount of adhesive injected).

V_m be the individual annual volume for model m ;

Formula 1 gives a set of values of T_p , one for every process. Next W_p , the number of workstations required per process p per shift is found:

$$W_p = \left\lceil \sum_{m \in M} T_p / (Wk \cdot Ds \cdot Hs \cdot 3600 / N_{sh}) \right\rceil \quad \dots \quad (2)$$

Subject to:

$$W_p \geq 1, \text{ and integer;}$$

Where:

- Wk is the number of weeks available per year;
- Ds is the number of days available per week;
- Hs is the number of hours available per shift;
- N_{sh} is the number of shifts available per day;

Formula 2 gives the number of workstations W_p required to assemble M models for all processes p . These workstations are process-specific and could be considered, at this stage, equivalent to one robot and one process (such as a welding gun).

The next operation is to create a process priority list. This is accomplished by summarising the joint database's preferences into strings. Thus, if a joint requires process a to be done before process b , but can do either process c or d in any order, then SIMAID creates an entry for this joint of: $[a-b, (c, d)]$. Example: Possible joints and their process preferences at phase 1:

<u>Joint</u>	<u>Order</u>	<u>Meaning</u>
J1:	3-4-7;	3, then 4, then 7;
J2:	2-1;	First 2, then 1;
J3:	5-(2,1)	5, then 2 and 1 in any order;
J4:	6-(3,4,8)	6, then 3, 4 and 8 in any order;
J5:	6-3-4.	6, then 3, then 4;

Table 4.1 Example of a subassembly requiring the processing of 5 different joints. The bracketing signifies the possibility of processing in any order.

Thus for j1, the preference would be to do process 3 first, then 4 and finally 7. The result is the following sequences:

- 3-4-7,
- 2-1,
- (5-2-1 or 5-1-2),
- (6-3-4-8 or 6-3-8-4 or 6-4-3-8 or 6-4-8-3 or 6-8-3-4 or 6-8-4-3),
- 6-3-4.

4.1.2 Phase 2: Creating a shortlist by sequence matching.

Phase 2 string-matches the sequences found in phase 1, starting with the greatest (longest) process sequence and attempting to match the shorter sequences, to the limit of 4 processes. The aim is to derive a few sequences that satisfy all of the subassemblies' processing and process priority requirements.

The longest sure ones are 3-4-7 and 6-3-4, and from these the string 6-3-4-7 ensues. This sequence satisfies the requirements for both joints. Out of the preferences for J4, the only one that would match here is the first possibility, 6-3-4-8. For J2 and J3 only one string could be selected, 5-2-1, which meets both requirements. Hence, at the end of Phase 2 there are 3 sequences, 6-3-4-7, 6-3-4-8 and 5-2-1.

4.1.3 Phase 3: Placing the requirements into Groups.

Phase 3 is to reduce or concatenate the process preference strings into unique non-repeatable sequences, beginning with those joints which have a preference and ending with those whose lack of preferences (if any) can be incorporated into the rest. If no matches are found, new sequences are formed to the least incumbent combination. By phase 3, depending on the number of different processes required, SIMAID creates a number of 'process groups' that will serve as the basis of cell formation. The number of such groups, N_g , is given by:

$$N_g = \left[(N_p / RC_{\max}) + \alpha + \beta \right]; \quad \dots \quad (3)$$

Where:

$$\alpha = 1 \text{ if } (N_p / RC_{\max}) < 1,$$

$$0 \text{ otherwise,}$$

$$\beta = 1 \text{ if } \sum_{g \in G} (N_{p,g} / RC_{\max}) = 1,$$

$$0 \text{ otherwise;}$$

Subject to:

$$N_p \geq N_g \geq 1, \text{ and integer;}$$

$$\sum_{g \in G} N_{p,g} \geq N_p;$$

Where:

g is a group of processes p ;

G : = {groups g present in the facility};

N_p is the total number of different process required in the facility;

$N_{p,g}$ is the number of different processes p present in group g .

RC_{max} is the maximum number of robots per cell (a user input);

The result of phase 3 is the creation of a certain number of groups with an allocation of processes within them. For example, if there are 8 processes then they are allocated into 3 groups. Fig 4.1 illustrates this:

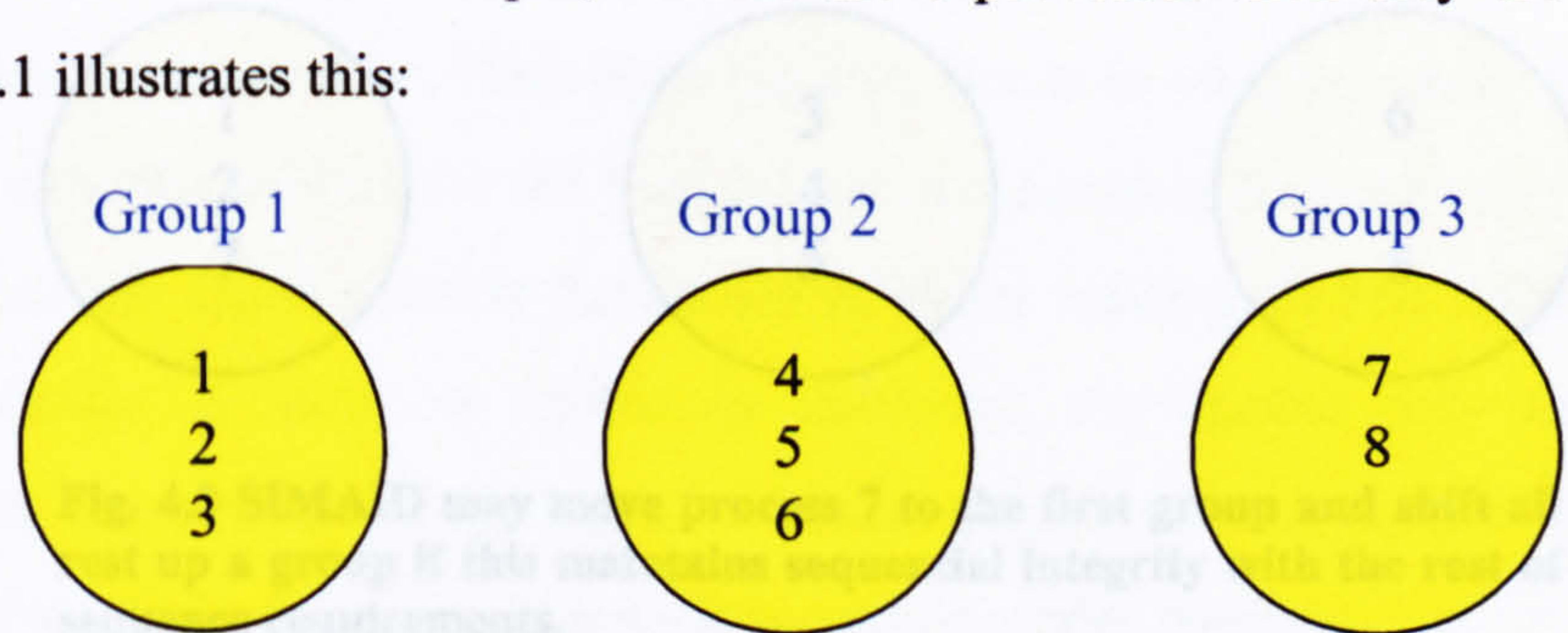


Fig. 4.1 SIMAID would place 8 processes into three groups as shown. The actual order of placement depends on the specific process preference sequences derived in the earlier phases.

The methodology involved in process placement is summarised in section. 3.31. However, occasionally the process preferences may not lead to such a simple placement, as there may be different subassemblies with contradictory requirements. For example, if subassembly A requires the use of process 2 after process 1, but subassembly B requires the use of process 1 after process 7, this may confuse the program into either placing workstations of process 1 into both Groups 1 and 3, or, depending on which subassembly is listed first in the database, placing both into stage 3. Fig 4.2 illustrates this point.

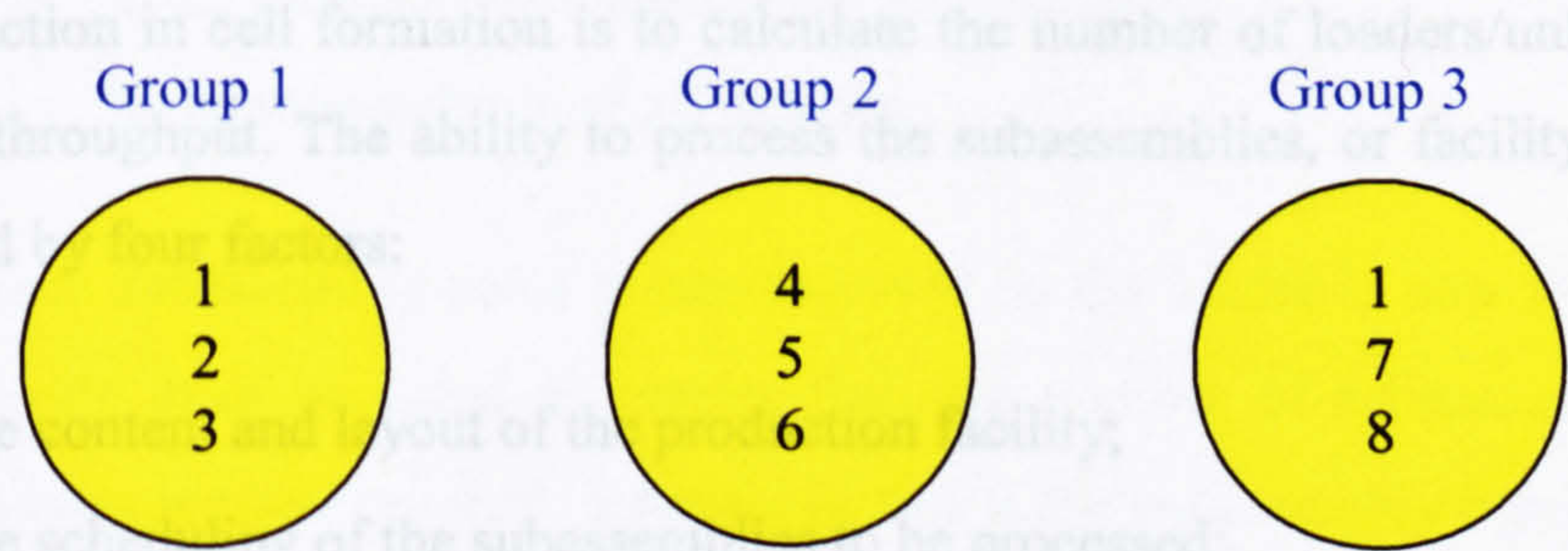


Fig. 4.2 SIMAID may duplicate process 1, placing it in both groups 1 and 3.

Neither solution is particularly adequate but unfortunately such situations do occur within industry. The resulting facility design would include unnecessary process and/or cell duplication, with the cost and space penalties this implies. To try to avoid this, SIMAID has a built-in 'safety loop' which attempts to get around this problem, if possible. The exact manner of operation is described in pseudocode in Appendix 4A, but in essence it tries to move some processes either up or down the groups, while maintaining sequential integrity. Thus, in the above situation, a possible outcome (depending on the rest of the sequence priorities) could be moving process 7 to the first group, as seen in Fig. 4.3:

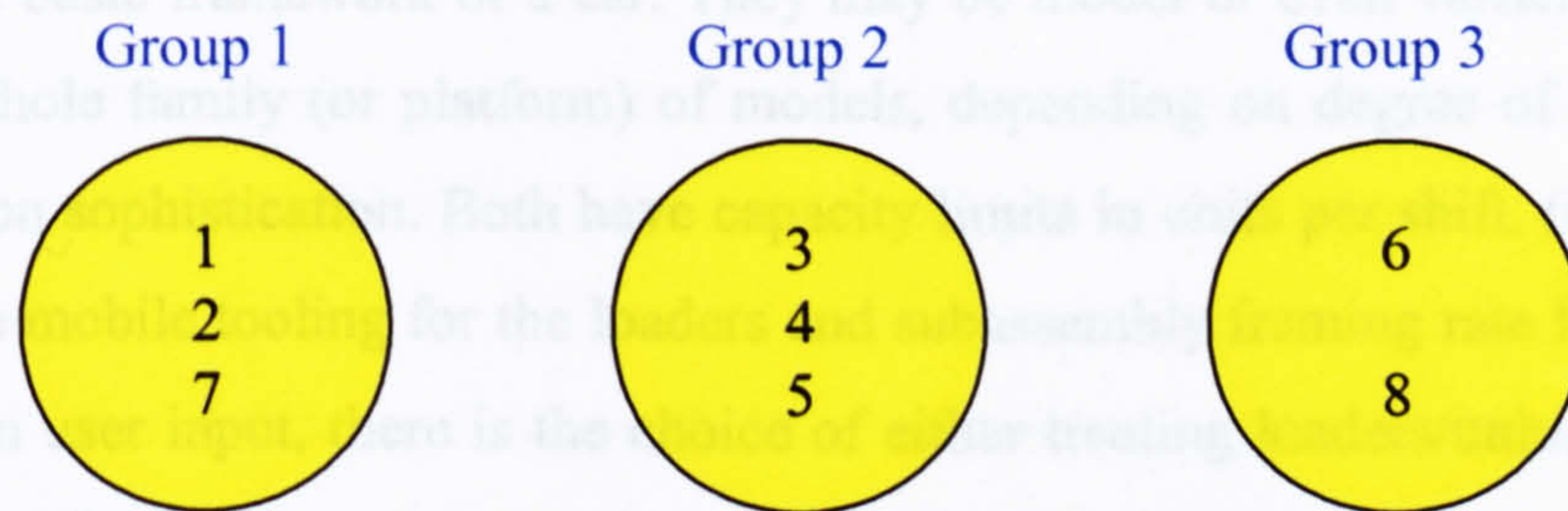


Fig. 4.3 SIMAID may move process 7 to the first group and shift all the rest up a group if this maintains sequential integrity with the rest of the sequence requirements.

From experience, the above loop is not often called, but if it is it results in a simpler group system. However, if the messages shown in appendix 4A do come up, this should be a warning to the user that either:

- A non-fatal error has been entered into the 'Process data' or 'Joint data' databases², encouraging a revision of the input data, or
- One or more of the subassemblies' process requirements are contradictory. Of course, this may be a correct requirement anyway.

4.1.4 Creating loaders/unloaders

The first action in cell formation is to calculate the number of loaders/unloaders required for the given throughput. The ability to process the subassemblies, or facility processing rate, is determined by four factors:

- The content and layout of the production facility;
- The scheduling of the subassemblies to be processed;

- The subassembly components' loading rate onto the mobile tooling, and
- The subassemblies' unloading from the tooling and framing rate after processing.

In the two subsequent stages, the methodology provides for both the iterative design of the facility content and layout to accommodate these production requirements, and for the subassemblies' scheduling requirements. However, before this iterative process begins, SIMAID needs to create the initial layout, complete with the adequate number of loaders and unloaders. As mentioned earlier, the term 'unloader' used here is equivalent, in the automotive industry, to framing stations that collect the relevant subassemblies and assemble them into the basic framework of a car. They may be model or even variant-specific, or may cater for a whole family (or platform) of models, depending on degree of parts-sharing and framing station sophistication. Both have capacity limits in units per shift, (i.e. subassemblies loaded on the mobile tooling for the loaders and subassembly framing rate for the unloaders). Depending on user input, there is the choice of either treating loaders/unloaders as universal or model-specific. If they are treated as universal, this means they can operate on all subassemblies irrespective of type, size or processing requirements, if otherwise, a separate loader and/or unloader is required for each model or model family.

First the mean weighted number of components per subassembly is found:

$$CS_{mw} = \left[\left(\sum_{m \in M} \sum_{s \in S_m} (N_s \cdot V_m) / S_m \right) / V_t \right] \quad \dots \quad (4)$$

Where:

CS_{mw}	is the mean weighted number of components per subassembly;
N_s	is the number of components per subassembly s ;
V_m	is the volume per model m to be built;
S_m	is the number of subassemblies per model m ;
V_t	is the total volume of all models;

The time each subassembly takes to be processed by the loader is now required. The loader-processing rate must be equal or greater than the number of subassemblies of all models required to be processed per shift. Essentially, this is the subassembly component loading time

² SIMAID checks for all 'fatal' errors, i.e. those which are likely to crash the program or make nonsense of the grouping system at a much earlier stage, when pressing the 'Update database' button.

plus the time the mobile tooling takes to move out of the loading cell, making way for the next one. The volume of units of all models required to be processed per shift, V_s , is given as:

$$V_s = \sum_{m \in M} (V_m / (W_k \cdot D_s \cdot N_{sh})) \quad \dots \quad (5)$$

The total amount of time required to load every component per shift, T_L , is:

$$T_L = \sum_{m \in M} \sum_{s \in Sm} ((T_{cl} \cdot CS_{mw}) + T_{tl}) \cdot V_s \quad \dots \quad (6)$$

Where:

T_{cl} is the average component loading time, in seconds, per component;

T_{tl} is the time taken for the mobile tooling to leave the loader;

In equation 6 above the time taken to leave the loader depends on tooling transporter acceleration and the minimum distance necessary to enable the next tooling to enter the loader. This is usually the length of the tooling itself. However, these characteristics are intrinsic of the MHS used and accurate values difficult to find. SIMAID does not provide user input for T_{tl} , but this feature could easily be implemented if required. For the purposes of the program, equation 6 is thus simplified to:

$$T_L = \sum_{m \in M} \sum_{s \in Sm} (T_{cl} \cdot CS_{mw}) \cdot V_s \quad \dots \quad (7)$$

The time available for manufacture, T_m , in seconds per shift, is given as:

$$T_m = ((H_s \cdot 3600) + (Minutes \cdot 60)) \quad \dots \quad (8)$$

From this the number of loaders (L_n) is found:

$$L_n = \left\lceil T_L / T_m \right\rceil \quad \dots \quad (9)$$

Unloaders/framing stations may also be either model or non model-specific. If each set of subassemblies making up a model requires its own framing station, then there may be more than one unloader in the facility. The number of such unloaders can be calculated by first finding the number of subassemblies required to be framed together:

$$V_{sm} = V_m / N_{sf} \quad \dots (10)$$

Where:

V_{sm} is the volume of subassemblies required to be framed per model;

N_{sf} is the number of subassemblies required to be framed together;

N_{sf} usually corresponds to the number of subassemblies in a model. V_{sm} is in units per year. To find the number required to frame in an hour, the number of hours of production in a year, H_{py} , is required:

$$H_{py} = [(W_k \bullet D_s \bullet H_s) + (\text{Minutes}/60)] \quad \dots (11)$$

The number of subassemblies of a specific model required to be framed together per hour, N_{sfh} , is thus:

$$N_{sfh} = V_{sm} / H_{py} \quad \dots (12)$$

Finally, the numbers of unloaders/framing stations required per model, U_n , is:

$$U_n = \left\lceil N_{sfh} / R_{fr_{max}} \right\rceil \quad \dots (13)$$

Where:

U_n is the number of unloaders/framing stations required;

$R_{fr_{max}}$ is the framing station's maximum framing rate, in units per hour;

If however the same unloader can be used for every model, then formula 10 should still be calculated per model, as each model may have a different number of subassemblies in it. However, equation 12 needs to be modified to sum all models, as such:

$$N_{sfh} = \sum_{m \in M} (V_{sm} / H_{py}) \quad \dots (14)$$

As the framing rate will be identical for all models, U_n can then be found normally using equation 13.

4.1.5 Cell creation.

The next phase of cell formation within this stage is the creation of the cells in this initial layout, consisting of cell creation, process allocation and positioning. The cells are created by first counting the number of workstations required for each process, and then allocating these to cells according to the Group they're in. The actual number of cells created in the facility, N_{cf} , is given by:

$$N_{cf} = \sum_{g \in G} \text{Max} \{W_g\} \quad \dots (15)$$

Subject to:

$$(N_{cg} \cdot RC_{max}) \geq W_{t,g} \geq \text{Max} \{W_g\} \geq W_{p,g} \geq 0;$$

Where:

- $\text{Max} \{W_g\}$ is the largest number of workstations of any process present in group g ;
- $W_{t,g}$ is the total number of workstations of all processes present in group g ;
- $W_{p,g}$ is the number of workstations of process p present in group g ;
- N_{cg} is the number of cells in group g ;

N_{cf} cells are created, each potentially capable of housing up to RC_{max} robot/process combinations. The RC_{max} figure recommended for SIMAID is four, considering the unfeasibility of having more than four robots attempting to operate simultaneously on a subassembly, which would result in robotic arm interference, collision risks and longer total processing times. However, this figure, a user input, may be changed if different robot configurations become possible. Finally, the last phase of Stage 1 is cell formation itself.

4.1.6 Facilitising with the data.

The workstations are initially given a cell each, starting with process p_1 to p_n , with new cells being created when the current cell number is less than the current required process number, and subsequent processes are added to the same cells, starting from C1, within the same stage until all workstations are accounted for. This is repeated for all Groups, until a first layout is obtained, with the first cell positioned at one corner of the available space, and the rest added below that in an orderly fashion, with subsequent Groups' cells placed directly behind the first Group's, separated by the width of the MHS and/or the minimum required intercellular space. If a necessity occurs to create a cell when available facility width becomes a constraint, SIMAID creates a cell in the next row, behind the first one, and carries on populating it, and

subsequent cells, with workstations. New cells from the next Group are not, however, added below this one, but cell construction begins behind the first cell of the last column of the last Group.

Example: Let's assume that for a hypothetical facility SIMAID has evaluated the requirements of the inputted data in the following manner:

- Six processes are required, processes 1-6;
- The preferences are in the order 1, then 2, then 3, then 4, then 5, and finally 6;
- Three stages are more than sufficient for six processes, which are placed two to a stage;
- From the above methodology, four workstations are required for process 1, three each for processes 2 and 3, two for process 4 and one each for processes 5 and 6;
- The number of loaders/unloaders for this example is assumed to be one each;
- The space available to facilitate is sufficient in width and length to fit the cells without resorting to double columns;

Fig. 4.4 below illustrates the methodology.

Workstations:

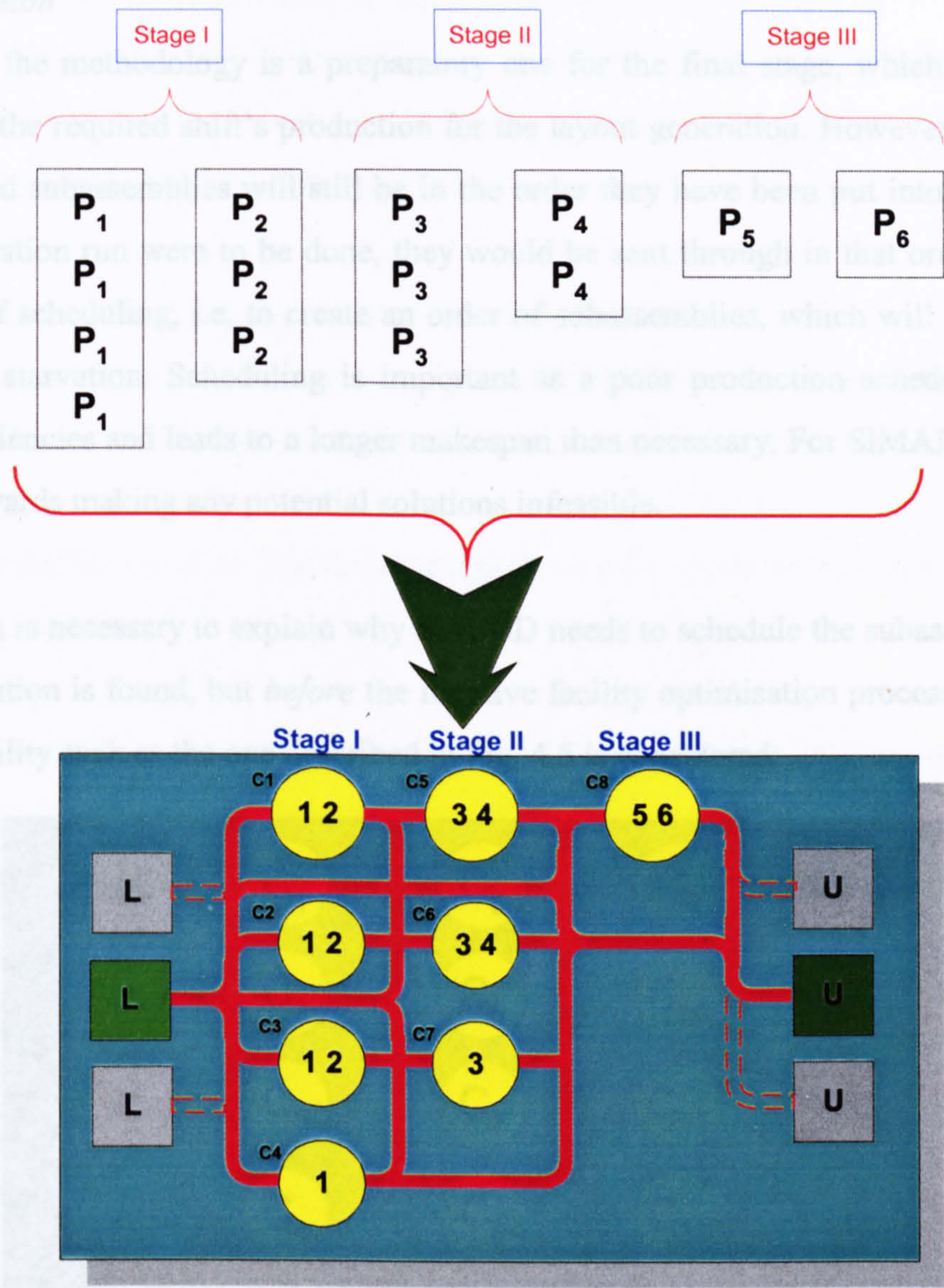


Fig. 4.4 With the information shown in the top part, SIMAID designs the initial facility as shown above (blue zone). The three stages described above do not represent the three stages of the methodology, but rather three processing stages the subassemblies need to go through in the facility. The method is: The greatest number of workstations are from process 1, hence a cell is created for each, starting from the top left hand corner (cell 1 –C1), and working down. Process 2 workstations are populated in the available cells as required, and the same principle is applied to the rest, each in the appropriate stage, in sequence. Each cell should be independently accessible from the loader(s), shown in light green, and to the unloader, shown in dark green. The red lines are possible MHS paths (SIMAID does not draw them, but assumes their presence for the purpose of routing. The grey boxes are other potential loaders and unloaders, if necessary.

4.2 Stage II

4.2.1 Introduction

This stage of the methodology is a preparatory one for the final stage, which relies on the simulation of the required shift's production for the layout generation. However, at this stage the models and subassemblies will still be in the order they have been put into the database, and if a simulation run were to be done, they would be sent through in that order. The issue here is thus of scheduling, i.e. to create an order of subassemblies, which will minimise cell blocking and starvation. Scheduling is important as a poor production schedule inevitably creates inefficiencies and leads to a longer makespan than necessary. For SIMAID, it will also contribute towards making any potential solutions infeasible.

At this point it is necessary to explain why SIMAID needs to schedule the subassemblies after the initial solution is found, but *before* the iterative facility optimisation process commences. Suppose a facility such as the one described in Fig. 4.5 is considered:

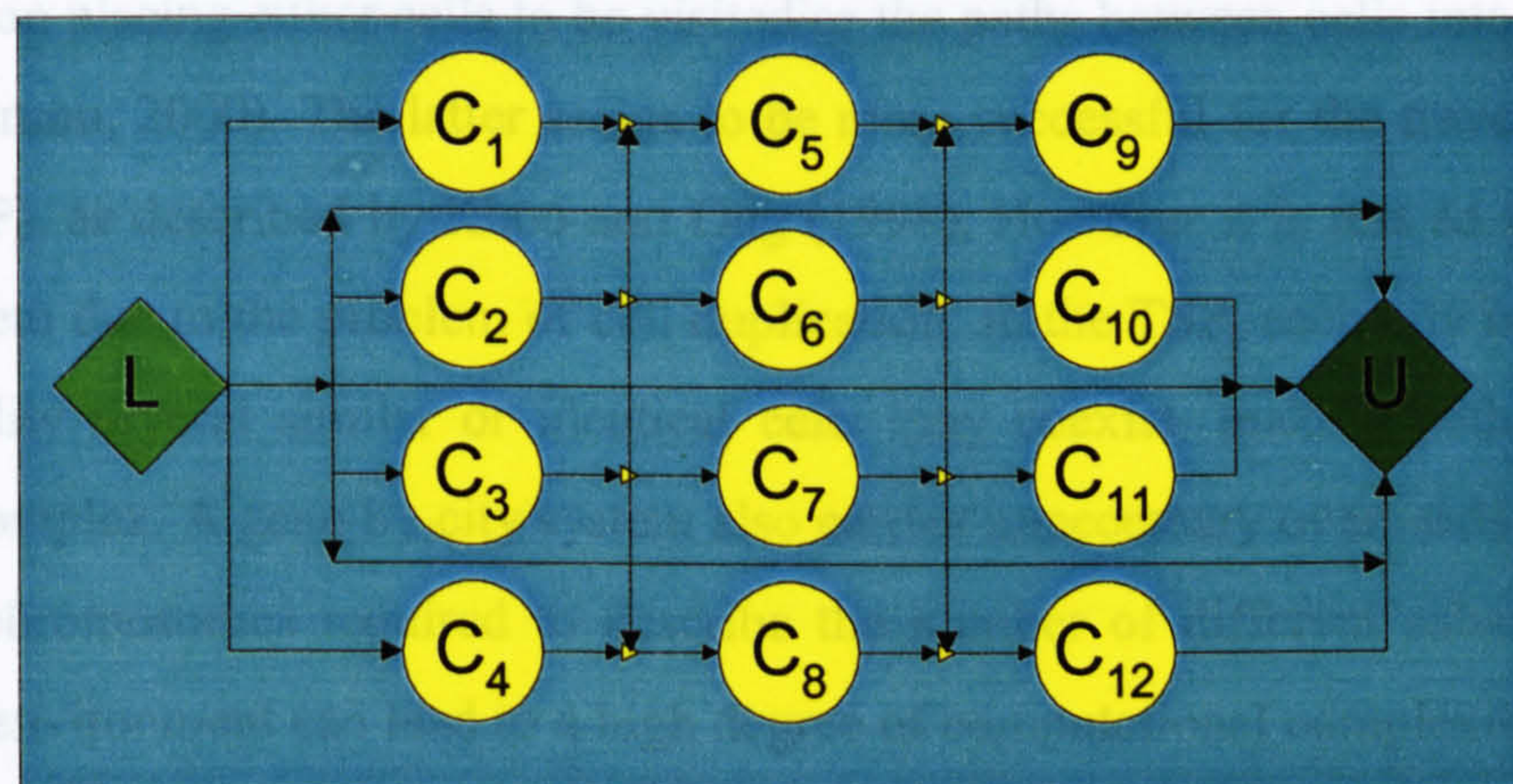


Fig. 4.5 A hypothetical facility design.

If, for example, there are 10 assemblies in a model that require n processes passing from the loader (L) to the unloader (U). Each column in Fig. 4.5 shall be called a processing, or manufacturing stage. This means that if the subassembly requires one or more processes present in any cell in that column, it then requires stopping within that stage. Typically, the units leaving the unloader may pass through one (and only one) of the cells C1 - C4, then may pass onto any one of the next stage's, C5 - C8, and finally may pass through any one of cells C9 - C12. However, not all of these subassemblies may require any or all the processes present in a specific stage. A subassembly may not necessarily require the type of processes present, for example, in any of the Stage 2 cells, C5 - C8, thus going directly to the third stage.

The aim of scheduling the subassemblies is to get the least possible throughput time (shortest makespan), given the existing (first, or initial) cell layout. Scheduling in a mixed-model, multi-process and parallel cell environment creates certain challenges not readily solved by conventional line or single-product scheduling, as closed form expressions are not easily constructed. The problem can be stated as follows: Given:

- A matrix of cells c_1, c_2, \dots, C , where C is the total number of cells,
- Each containing a variable number of potentially different processes p_1, p_2, \dots, P , and
- A set of subassemblies s of m different models, each of different volumes v ;

How do you schedule them so that the predetermined daily quota of cars can be assembled within 1,2 or 3 shifts?

In SIMAID scheduling is carried out using genetic algorithms (GAs). Past efforts have concentrated on placing either cells to be visited or the paths between cells into chromosomes (Wang and Brunn, 2000). The latter seems to be more successful for the travelling salesman problem (TSP), as described by Khoo and Ong (1998). However it is less so for the parallel facility problem due to the problem of cell duplication. In the TSP, each city is unique but in an agile facility several similar or identical cells may coexist, making path incorporation necessarily complex. A gene-by-city system also carries unnecessary overheads as the number of different chromosomes required to describe the number of different subassemblies in a multi-model environment can lead to a high degree of computational complexity. In this work it is neither paths nor 'cities', but the subassemblies themselves which are incorporated as the genes of the chromosomes. A chromosome, therefore, is a sequence of subassemblies to be processed through the current facility.

After a suitably sized population of randomly generated chromosomes are created, GA algorithms (SIMAID's included) measure their performance under a set of criteria and order them according to which ones best fit them. Typically, the chromosome list order changes from the randomly generated one at the beginning, to one in which the first in the list describes the current best solution, in this case, the shortest makespan. The next step typically consists of three distinct parts, selection, reproduction and mutation.

1. *Selection* - Individuals are selected for reproduction. The selection can be random, where the probability of being selected is proportional to their fitness value, deterministic, usually selecting a small percentage of the 'best performing' chromosomes, or a mixture of both, such as enabling the reproduction only between randomly selected pairs from the top (better) half of the table.
2. *Reproduction* - The genes of the two selected chromosomes are recombined to create one or more new one by crossover. Both the actual point of crossover and the number of crossover points can be varied.
3. *Mutation* - A single gene can be mutated in any chromosome to represent something else. This is usually done to avoid the situation where a local maximum is reached and the GA stops there, believing it has reached the global maximum instead.

GAs then re-iterate a pre-set number of times, improving the target value until no more improvements are detected. The best solution is then selected as the answer.

4.2.2 The measurement of fitness

SIMAID follows the above GA principles in creating the population of chromosomes, allowing for reproduction and executing improvement selection iterations. As each chromosome represents a schedule, the aim is to achieve the shortest makespan. Thus, in this case the 'fittest' (best) chromosome is the one that can process the given set of subassemblies through the facility in the shortest possible time.

There are two possible approaches to the issue of its measurement, quantitative or qualitative. The former involves calculating the chromosome (schedule) processing completion time from processing requirements and travel time, measuring the difference between the time of the first subassembly leaving the loader to the last one entering the unloader. There are a number of qualitative approaches, one of which is using simulation to measure the time.

SIMAID evaluates completion time with simulation, rather than calculation. The reason for this is that in an agile, parallel facility, unlike with conventional lines, not only can several subassemblies be processed simultaneously in the facility (making processing time calculations difficult), but more than one subassembly can also be processed in a cell, such as

a marriage cell. Additionally, cell availability at any stage of the exercise is unknown. In a conventional production facility, which has a predetermined cycle time, it is relatively simple to evaluate the completion time for any job (such as a production batch). By contrast, in a parallel cellular facility each cell could service many types of subassembly, each with different processing requirements and hence completion times. As these subassemblies are sent to the first available (and compatible) cell in real time, which could also be situated at varying distances from the loader or original cell, a precise calculation of total job completion time becomes complex. Finally, even if feasible, the computing cost in terms of time taken to find the solution to this type of problem becomes prohibitive.

The task of finding the chromosome (schedule) with the shortest makespan, given the current facility cell configuration, is done by simulation. Each chromosome is sent through the current facility and the time taken recorded. After reproduction, the exercise is repeated with the new set of chromosomes, each time attempting to minimise the chromosome completion time. The function can be described as:

$$\text{Min}(T_{\text{comp}}(c) \dots \forall c \in P) \dots (16)$$

Where:

T_{comp} is the completion time of chromosome c ;

c is a chromosome (schedule) containing a sequence of genes (subassemblies);

P is a population of chromosomes c ;

4.2.3 The prototype SIMAID scheduling GA

The SIMAID scheduling GA has several sequential steps to prepare the ground for the simulation runs. The usual GA steps are required, such as gene selection, chromosome incorporation, determination of the size of the population, determination of the number of generations (improvement iterations), selection of the method of reproduction, determination of the number of offspring, and finally the selection of an appropriate mutation rate. Although many of these functions are standard to all GAs, their usage in scheduling subassemblies in an agile, parallel, cellular facility brings its own challenges that require certain modifications to ensure functionality. Fig 4.6 gives an overview of the process:

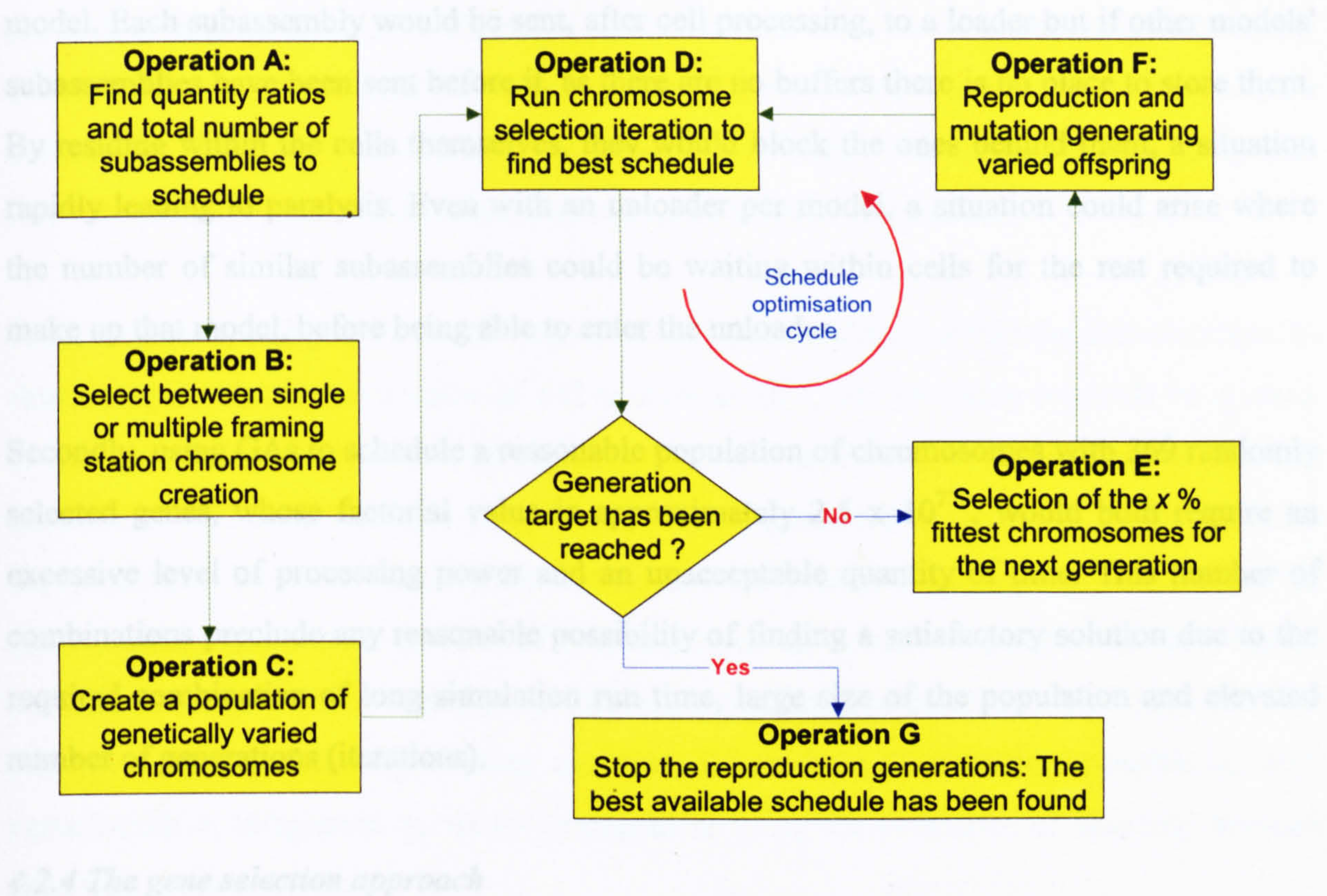


Fig. 4.6 Flowchart of the second stage of the SIMAID methodology, which uses raw subassembly and volume data to generate good (sub-optimal) schedules.

The concept is best understood with an example. Assume there are 4 models that need to be scheduled, with 48 weeks x 5 days x 2 x 8 (3840) hours of available time:

Model	Annual volume	Per hour	Ratio	Subassemblies	Total (approx.)
Model 1	50,000	13.02	5	14	70
Model 2	20,000	5.21	2	12	24
Model 3	40,000	10.42	4	10	40
Model 4	10,000	2.6	1	8	8

Table 4.2 A hypothetical production set.

A shift's worth of production (8 hours) would require to schedule $((13 \times 14) + (5.2 \times 12) + (10.4 \times 10) + (2.6 \times 8)) = \sim 369$ subassemblies. GAs normally generate chromosomes by concatenating the genes (in this case the subassemblies) selected randomly. However, this is not practical for two reasons. Firstly, if the number of unloaders/framing stations for all models is less than the number of models, this will create a problem. There could be, theoretically, up to 182 (13 x 14) subassemblies of model 1 in between any two of any other

model. Each subassembly would be sent, after cell processing, to a loader but if other models' subassemblies have been sent before it, as there are no buffers there is no place to store them. By residing within the cells themselves, they would block the ones behind them, a situation rapidly leading to paralysis. Even with an unloader per model, a situation could arise where the number of similar subassemblies could be waiting within cells for the rest required to make up that model, before being able to enter the unloader.

Secondly, using GAs to schedule a reasonable population of chromosomes with 369 randomly selected genes, whose factorial value is approximately 2.5×10^{778} , would both require an excessive level of processing power and an unacceptable quantity of time. This number of combinations preclude any reasonable possibility of finding a satisfactory solution due to the required combination of long simulation run time, large size of the population and elevated number of generations (iterations).

4.2.4 The gene selection approach

The SIMAID GA proposed here uses a more rational, mixed approach. First, we find the lowest whole number common denominator for both the models' subassembly number and their respective volumes. The formula to find the minimum number of genes per chromosome is:

$$\sigma = \text{Min} \left(\sum_{m \in M} N_{sm} \cdot N_{IMR} \right) \quad \dots (17)$$

Subject to:

$$\begin{aligned} N_{IMR} / N_{sm} &= \text{Whole multiples of } N_{sm}; \\ \sigma &= \text{Whole multiples of } m \dots \forall m \in M; \end{aligned}$$

Where:

- σ is the number of genes per chromosome;
- N_{sm} is the number of subassemblies s per model m ;
- N_{IMR} is the *individual model ratio*, in terms of production volume;

Since the subassembly ratio in the example is approximately 7:6:5:4, all of the models would be successfully built if the subassemblies were to be sent in those proportions. However, the volumes ratios are 5:2:4:1. This means that 5 vehicles of model 1 need to be built for each of model 4. Consequently, taking both volume and unit number into account creates an adjusted

production ratio of 35:12:20:4. However, if sent on their own, the 35 subassemblies of model 1 would only be able to build 2.5 vehicles, clearly unacceptable. Thus, a chromosome with a subassembly ratio of 70:24:40:8 is the minimum structure that could be repeatedly sent through the facility and still be consistent with production requirements.

From the above, 142 (70 + 24 + 40 + 8) subassemblies are placed within the chromosomes, in this example. The factorial value of 142 is 2.69×10^{245} , but in reality it would be a much lower figure. The reason is that, as explained above, it would be potentially fatal to expect a schedule of 142 subassemblies placed in randomly generated order to be viable. Identical subassemblies (from the same model, but different individual vehicles) can be sent into the facility before the correct number of subassemblies have been despatched of the original vehicle, leading to an occurrence of the notorious facility paralysis. The SIMAID GA circumvents this problem by creating, as shown, the smallest chromosomes possible but with certain built-in safeguards, primarily among them being the avoidance of inserting repeated (identical) subassemblies.

4.2.5 Chromosome formation

After the number of genes has been selected using equation 17, the actual order of the genes within the chromosomes needs to be determined. Following a sequential set of rules very early in the chromosome-creation process comprises the methodology for 'safe' chromosome creation. The idea is to allow as much gene sequence variation in the chromosome as possible without letting through 'stillborn' candidates. Unfortunately, the same methods cannot be applied to situations where a single unloader/framing station type can serve every model and where multiple, model-specific framing stations are necessary. In the former case scheduling is restricted to the formation of chromosomes comprised of whole subassembly sequences, as only subassemblies from a single model can be framed together. Hence, the gene sequencing within the chromosome is reduced to a combination of subassembly sequence variation within the set of subassemblies that constitute a model, and whole set sequence variation:

$$G_c = \aleph \{ \text{Rand} \{ \aleph \{ \text{Rand} \{ s \} \dots \forall s \in m \} \} \dots \forall v \in S_v \} \quad \dots (18)$$

Where:

G_c $:= \{ \text{genes in the chromosome } c \};$

- v := {subassemblies of a model comprising a whole vehicle};
 S_v := {vehicles selected for inclusion into the chromosome};
 $\bowtie \{ \}$ is the concatenation of elements (or sets of elements) present in set $\{ \dots \}$;
 $\text{Rand}\{s\}$ is the randomised selection of the element s from the set $\{ \dots \}$;

The sequence thus is the result of the concatenation of the randomisation of the concatenated randomised selection of the subassemblies from the set that constitute a model for all such sets of a model, for the set of such whole vehicles determined for inclusion into the chromosome. To explain better, using the above example, s would be the subassemblies of a model, which, for model 4 number 8. A randomisation of these subassemblies could create 40,320 possible combinations. Next, the same has to be repeated for all such sets, which for model 1 have been determined (from the volume ratios) as being 5, two for model 2, four for model 3 and one for model 4. Once the order has been determined, it is fixed and incorporated as genes. It is important to realise that these orders of genes are concatenated together, not mixed.

Where multiple framing stations are required, the problem domain is different. The level of cell blockage is considerably lower because, unlike in a single-sink situation, the subassemblies have the possibility of entering their respective unloaders when their processing requirements have finished. Hence the strict whole vehicle set criterion described in equation 18 does not necessarily apply, although the validity of the approach does. Consequently, with multiple framing stations the degree of chromosome variation can be increased by being able to mix the genes representing the different models more freely. However, facility paralysis can still ensue when one or more instances of two subassemblies of the same model are being processes in the facility simultaneously. Hence, equation 18 needs to be both simplified and constrained:

$$G_c = \bowtie \{ \text{Rand}\{s\} \dots \forall s \in S_s \} \quad \dots (19)$$

Subject to:

$$\Delta \{s_i, s_j\} \geq \eta \{ v \}; \quad \dots (20)$$

Where:

S_s	$:=\{\text{subassemblies of all models selected for inclusion into the chromosome}\}$
$\Delta \{x, y\}$	is the difference in positional distance, in terms of number of elements, between the two elements x and y in the set;
s_i, s_j	are two elements (instances), i and j of subassembly s in a set;
η	is the cardinality (number of elements) of set $\{ \dots \}$;

Equation 19 translates to: The set (and sequence) of genes in chromosome c is the result of the concatenation of the random selection of subassemblies s from the set containing all of the subassemblies (of all models) destined in chromosome c . The constraint 20 means that the distance between two identical subassemblies from different instances of the same model must be at least equal to the number of subassemblies in that model.

Using the model as described in fig. 4.6, the example can be followed through:

1. *Operation A: Find quantity ratios and total number of subassemblies to schedule.* In this case, the models have subassembly abundance quantity ratios of 70:24:40:8 and as seen in section 4.2.4, 142 units need to be scheduled, the chromosome must, therefore, contain 142 genes.
2. *Operation B: Select between single or multiple F.S. chromosome creation.* If the assembly system allows a single unloader/framing station for all models, use formula 18 to arrange the genes in the chromosome; if one per model is required, then use formula 19.
3. *Operation C: Create a population of genetically varied chromosomes.* From section 4.2.5, a population of chromosomes is generated. Each gene, representing a subassembly, carries information stating only which *processes* it requires and their duration, but without naming any cells. Let's say that a portion of 8 subassemblies require the processes shown below:

$S_1 = 12468$	$S_5 = 12468$
$S_2 = 13568$	$S_6 = 24678$
$S_3 = 35678$	$S_7 = 13568$
$S_4 = 15678$	$S_8 = 23568$

S_1 requires processes 1,2,4,6 and 8, but before actually testing it is not known whether it should be placed before S_2 , or between S_3 and S_4 , after S_7 , etc. As there could be any number of cells with those processes in each group, putting several subassemblies with starting requirements of P_1 , such as S_1 , S_2 , S_4 , S_5 , and S_7 , may lead to cell blocking in stage 1 and cell starvation in the subsequent stages, or even other cells in the same stage, creating a totally non-optimal schedule. On the other hand, such an inconvenience may be preferable if the processing times of P_1 are very short and the real bottleneck is at stage 3. Randomisation of the order of the subassemblies creates a population of sequenced chromosomes such as shown below:

$Y_1:$ $S_5 S_6 S_7 S_2 S_4 S_8 S_1 S_3$
 $Y_2:$ $S_6 S_7 S_5 S_4 S_1 S_3 S_2 S_8$
 $Y_3:$ $S_6 S_5 S_2 S_7 S_8 S_1 S_4 S_3$
 $Y_4:$ $S_2 S_6 S_3 S_7 S_4 S_8 S_1 S_5$
 $Y_5:$ $S_8 S_1 S_5 S_7 S_2 S_4 S_3 S_6$
 $Y_6:$ $S_2 S_4 S_7 S_3 S_5 S_6 S_8 S_1$ etc.

The population size is important as it allows for greater initial genome variability. The size of the population should be as large as possible in order to give the program a chance of having at least an average set of genes in any chromosome, but large populations incur long processing times. The default level set by SIMAID, found to be adequate by experimentation, is proportional to the size of the chromosomes:

$$SZ_{POP} = 10-50 \sigma \quad \dots (21)$$

Where:

SZ_{POP} is the population size;

The variation in the population size multiple ($10-50\sigma$) is due to relative processing times. For a small chromosome of 20 genes, a population of 1000 is adequate, but if the genome size were 142 like in the example, this would mean a population of 7100 chromosomes, with the corresponding time penalty. It is clearly desirable to put a limit in the form of a user-controlled sliding scale.

Resulting in:

4. Operation D: *Run chromosome selection iteration to find best schedule*. The population of chromosomes is simulated, $2n$ units at a time (where n is the number of cells in the current facility layout, as described in section 3.4.2), and after processing and reconstituting are placed in makespan rank, with the chromosome with the shortest makespan as the 'fittest', followed by the next fittest, etc. If this is the first generation, then the next step is operation E. If the number of generations meets the target level, the control of the program leaves the scheduling stage and enters Stage 3. The best schedule (fittest chromosome) found so far constitutes the one used in the next stage.
5. Operation E: *Selection of the $x\%$ fittest chromosomes for the next generation*; the ranked chromosome table is divided into those allowed to participate in reproduction and those deleted from it. The former are then put forward for engendering offspring. The number of chromosomes selected varies according to mating policy and offspring fertility (number of children engendered), discussed in section 4.2.7.
6. Operation F: *Reproduction and mutation generating varied offspring*. The selected chromosomes are randomly paired and allowed to reproduce. The children replace the deleted chromosomes from the table and the cycle returns to operation D to check their makespan. This is discussed in section 4.2.6 below.

4.2.6 Reproduction

The number of children engendered per pair of chromosomes that could be experimented with, in this system, is up to 4. Consider the two parent chromosomes below:

Y_1 : 1 2 3 4 5 6 7 8 9 10 11 12

Y_2 : 9 3 6 10 2 7 4 5 1 12 8 11

The four children are created by the following single-site crossovers:

Y_1 : 1 2 3 4 5 6 7 8 9 10 11 12

↑ Crossover

Y_2 : 9 3 6 10 2 7 4 5 1 12 8 11

↑ Crossover

Resulting in:

Child 1: 1 2 3 4 5 6 9 10 7 12 8 11

Child 2: 9 3 6 10 2 7 1 4 5 8 11 12

Child 3: 3 6 2 4 5 1 7 8 9 10 11 12

Child 4: 2 3 6 7 9 10 4 5 1 12 8 11

Only non-repeatable sequences from each chromosome can be used, hence the complete halves of the chromosomes are complemented with a sequence consisting of non-already present genes from the opposite chromosome, in their original order. In this way full favourable gene sequence conservation is covered. Dual or triple-site genetic exchange is also possible, but as the site number increases so does gene 'confusion' and optimality convergence may not approach as rapidly, resulting in many more iterations required to achieve good results.

4.2.7 The mating mechanism

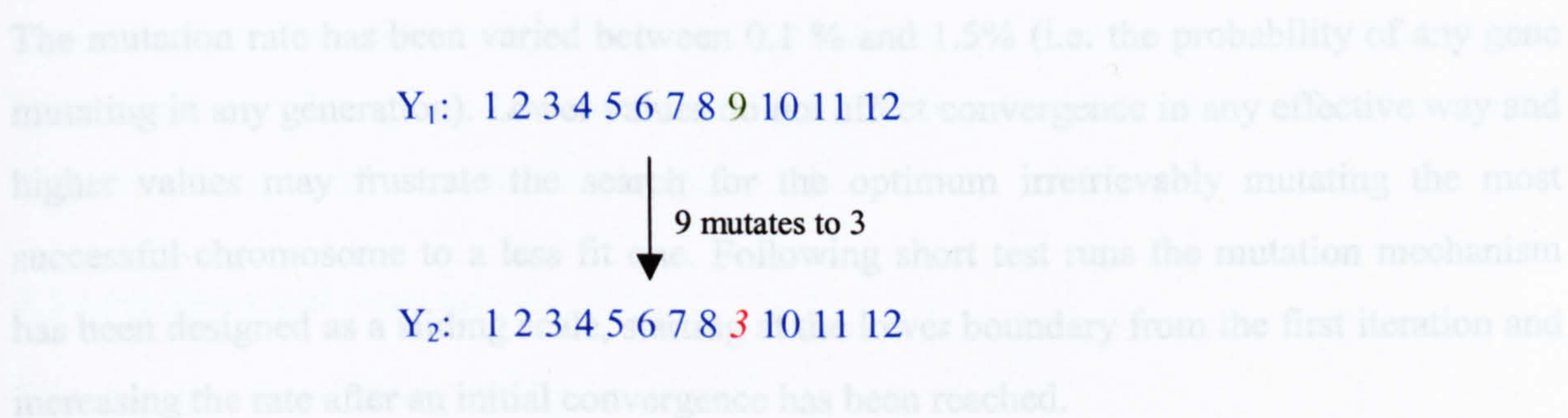
The mating mechanism used is a multiple monogamous one, where the best x % of the chromosomes are allowed to randomly mate with each other, each producing 2 children. The value of x % has been varied between 50% and 90%. Parents and children are then tested and ranked, and the worst x % is continuously deleted from the bottom of the table. A 2.5% average improvement has been found using 2 children instead of one, and further improvements are expected when 4 children are used. Subsequently, the crossover point has been randomly selected instead of using the nearest halfway point (e.g. after the first 6 genes in a 12-gene chromosome) for each mating. Polygamous mating mechanisms could also be used but have not been tested.

4.2.8 Mutation

In GAs, mutation is where a gene is randomly altered to another. This has beneficial effects due to the tendency otherwise of GAs to get stuck in a local maximum point when a higher value actually exists within the solution space. This tendency is more marked when the population sample is small in relation to the total number of possibilities. Without mutation, after a certain number of generations the population diversity decreases and the probability of localised convergence correspondingly increases. Mutation is thus used to mitigate this effect

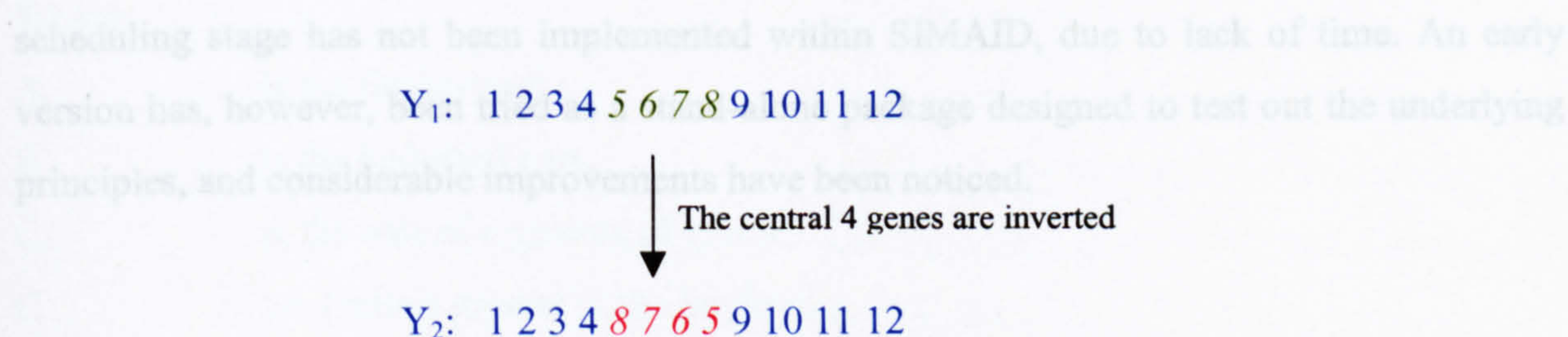
by introducing a certain degree of variation within the population, with the aim of enabling the 'leaping' out of any local convergence.

The crucial aspects of mutation are the mutation type and rate. Single-point mutation (*SPM*) is a typical method, where a specific gene is randomly selected to mutate to represent a different one. Although SPM is feasible in many fields, scheduling with such pre-set constraints (such as a number of subassemblies being required to enter the framing station to make a model) negates this possibility in SIMAID. A change of subassembly type will lead to similar problems as described in section 4.2.3: Facility paralysis could ensue if a model cannot be built due to a missing subassembly. If Y_1 describes a typical chromosome section, representing the subassemblies required to build a model, it can be seen that the mutated variant Y_2 has one missing and one extra, unneeded, subassembly:



Chromosome Y_2 may still be able to build the vehicle if by chance the next sequence happens to be from the same model, but there would still be a missing subassembly for a vehicle of this model; the problem is just postponed. Hence other mutation methods are used.

The types of mutations used are *inversion* and *transposition*. The former involves the random selection of a certain amount (2 genes +) of the genome and inverting its order:



Transposition involves the shifting of a randomly selected and variable-sized section of the genome to a neighbouring area:

4.3 Stage III

Y₁: 1 2 3 4 5 6 7 8 9 10 11 12

Green selection is shifted along in the genome sequence

Y₂: 1 2 6 7 8 9 10 3 4 5 11 12

It is worth noting that, for single framing station situations, these mutations, like with reproduction, can only be performed within those sections of the chromosome which have freedom to vary, i.e. within subassembly sequence runs belonging to the same model. Crossing over or mutating between models can only lead to problems. This restriction is much more relaxed with multi-framing station situations, where only constraint 20 need be observed.

The mutation rate has been varied between 0.1 % and 1.5% (i.e. the probability of any gene mutating in any generation). Lower values do not affect convergence in any effective way and higher values may frustrate the search for the optimum irretrievably mutating the most successful chromosome to a less fit one. Following short test runs the mutation mechanism has been designed as a sliding scale, starting at the lower boundary from the first iteration and increasing the rate after an initial convergence has been reached.

4.2.9 Iteration number

The minimum number of iterations required has been found to have a relationship with the gene size, being significant from approximately 4x gene size. Thus a 100-gene chromosome needs a minimum of 400 generations, but occasionally improvements have been found after as many as 3000 generations. The default SIMAID value has been set at 5x-gene size. The scheduling stage has not been implemented within SIMAID, due to lack of time. An early version has, however, been tried as a stand-alone package designed to test out the underlying principles, and considerable improvements have been noticed.

4.3 Stage III.

This is the final stage where iterative layout improvement is carried out to a specific set of objectives and constraints. It starts with an initial layout designed in stage I, and a set of scheduled subassemblies to be processed per shift, according to their relative models' demand. As this initial layout is infeasible, stepwise modification is required, approaching more feasible solutions. SIMAID doesn't know what the layout for a feasible solution looks like. As the number of subassemblies to be sent through the facility is varied through user input, the selected set's makespan is unknown. This is found by doing a first simulation run, i.e. all the subassemblies that need to be processed are pulled through the cells until the last one has reached the unloader. Experience shows that this can be anywhere between 150% - 300% of the required time. At this stage the first improvement iteration begins. SIMAID analyses this performance, identifies the biggest problem, and attempts, with a single change, to correct this. Then the second simulation run occurs, and it may (or may not) result in a better makespan. This process is repeated until a feasible solution is achieved, in other words, until the makespan matches the shift length or less.

4.3.1 Mathematical formulation.

The objective function is given by:

$$\text{Min } K = \left[\left(\sum_{p \in P} W_p \cdot C_p \right) + \left(\sum_{c \in C} \sum_{r \in R_r} \bullet \cdot C_{\text{rob},r} \right) + \varepsilon + \mu + \varphi + \gamma + C_b \right] \dots (22)$$

Subject to:

$$K \leq B;$$

Where,

K	is the facility cost ;
B	is the budgeted cost;
C_p	is the cost of a process of type p ;
C	$:= \{\text{cells } c \text{ present in the facility}\};$
R_r	$:= \{\text{robots of type } r \text{ present in a cell}\};$
$C_{\text{rob},r}$	is the cost of a robot of type r ;
ε	is the cost of the safety systems required per cell;
μ	is the total cost of the MHS utilised in the facility;

- φ is the total cost of the control systems employed in the facility;
 γ is the total cost of the area used for the facility;
 C_b is the total cost of the building, excluding above contents (green field only);

Equation (12) does not take into account the two major constraints, namely that the simulation run must be within the maximum time allowed, and that the cell area used must be smaller than the predetermined available space, which is a user input. The individual facility contents costs are arrived at as follows:

\mathcal{E} has two components: it has a basic cell value, depending on cell size, plus a part which is process-specific.

Let:

C_r := {cells c of size r in the facility}, where r corresponds to the actual number of robots present in that cell,

RC_{\max} ; be the maximum number of robots per cell,

$C_{s,r}$ be the cost of safety equipment of a cell of size r ,

$C_{s,p}$ be the cost of safety equipment of a workstation of process p ;

then,

$$\mathcal{E} = \left(\sum_{r=1}^{RC_{\max}} \sum_{c \in C_r} C_{s,r} \right) + (W_p \cdot C_{s,p}) \quad \dots \quad (23)$$

μ is the product of the total length of the MHS and its cost per unit length. This method is simplistic and more elaborate measures could be implemented, depending on the type of MHS that can be installed as well as the complexity of the model. SIMAID does not specify a type, but for the purposes of the paper we can assume it is a conveyor-type system that moves the tooling, holding the subassemblies, from the loaders to the unloaders through the cells. The cost of any X-type or Y-type junctions, which should be taken into account when inputting the initial value of the cost per unit length, has also been ignored.

φ is calculated in a similar manner to \mathcal{E} :

$$\varphi = \varphi_b + \left(\sum_{c \in C} \varphi_c \right) + \left(\sum_{g \in G} \varphi_p \cdot W_{p,g} \right) \quad \dots \quad (24)$$

where,

C := {cells c present in the facility} ;

φ_c is the cost of the cell-level control system of cell c ;

φ_b is a basic initial cost of the control system, whatever its size;

φ_p is the cost of the process-level control system of process p ;

$W_{p,g}$ be the number of workstations of process p present in group g ;

γ is calculated in a straightforward rectangular manner, from the size of the facility, by multiplying the length by the width of the facility to find the area, and multiplying the product against the cost of space, in £/Sq. metre. More comprehensive ways to calculate area could be used, including for irregular layouts, but these are beyond the scope of this work.

4.3.2 Simulation.

SIMAID has been incorporated with a control system which dynamically allocates the cell that each subassembly should be sent to, depending on its processing requirements, the state of the cell, and on whether, if the subassembly were to be sent to that particular cell, it could still satisfy all of its process requirements in the subsequent stages. Each stage has a specific set of processes, unique to the group, usually not present in any other group. The groups have been created so that a subassembly, starting from the loader(s) at one end of the facility, is pulled through each of the stages (if necessary) in turn until all of its process requirements are satisfied, without the need for backtracking.

It is necessary to understand, however, that there is no parts-process table where it is known where the subassembly should be next sent to, as with many other such algorithms. The SIMAID tooling router is *dynamic* - meaning it checks the subassembly's process requirements on loading and routes the tooling to the appropriate cell accordingly. Due to the issue of cell availability, the router learns which *type* of cells each subassembly needs to visit, but does not know which specific cell it will send it to, until it is its turn to leave the loader. This 'real-time' control system has several advantages, from the possibility of utilising the principles of late configuration, to cell breakdown damage limitation. The decisions reached by the controller can be seen in Fig 4.7 below:

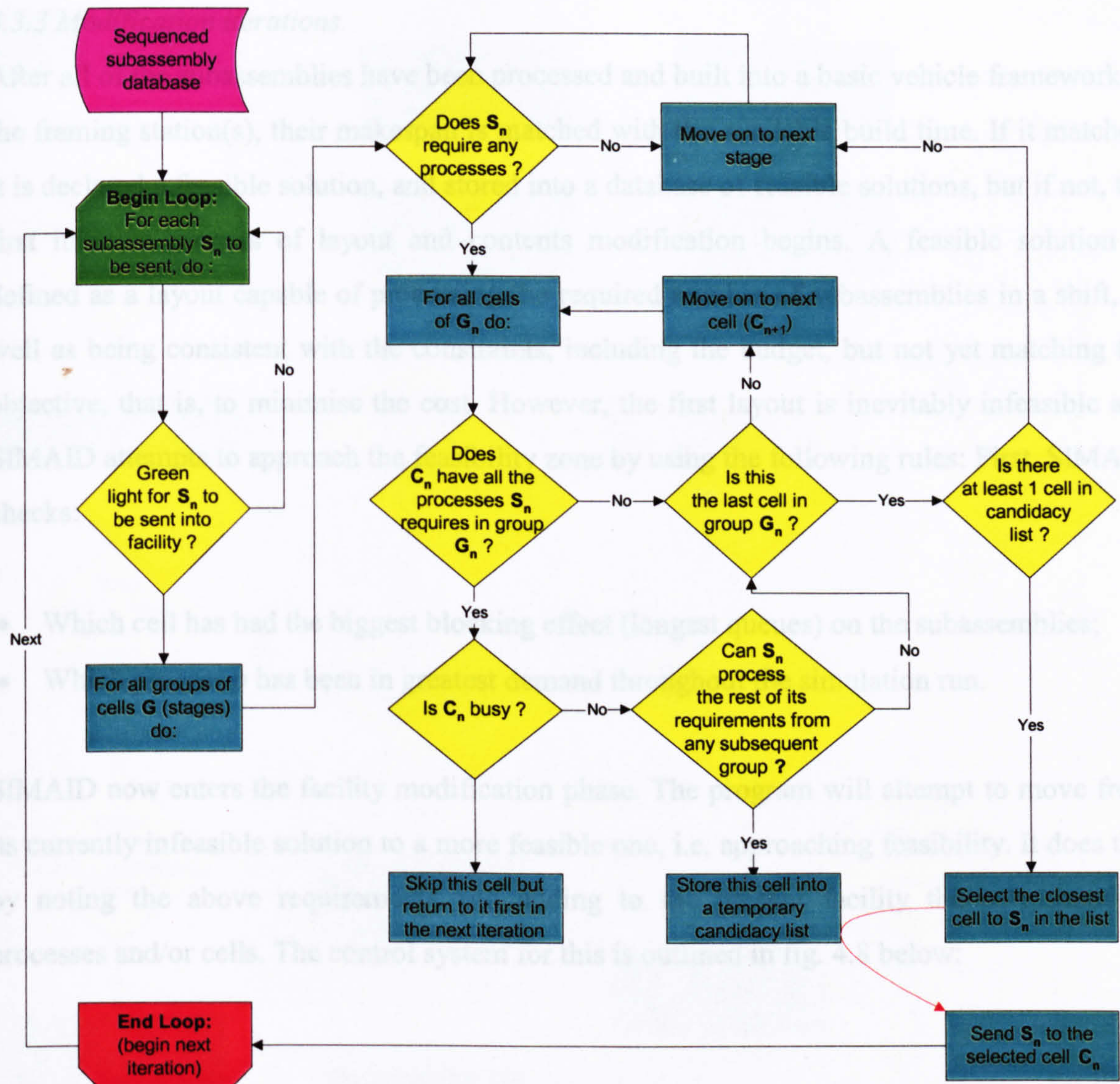


Fig 4.7 Subassembly routing despatch control.

This facility design is bufferless, and thus the subassemblies remain in their last entered cells until the next acceptable cell becomes available. An acceptable cell is defined as one which a) has the necessary processes required by the subassembly in question, and b) is currently free. This *pseudobuffer* method does not appear to be the most time-effective method, as the cells containing subassemblies are in effect blocking the entry of the subsequent ones either still at the loader(s) or at a previous stage. Even a 1-unit buffer, placed before or after each cell, or even a single, variable size buffer between the groups should improve the makespan at a (relatively) low cost, though penalising space. As none of these options have been incorporated into SIMAID, there is a correspondingly heavy reliance on scheduling. The first simulation run terminates when all of the subassemblies have had their processing requirements satisfied, and have reached the unloader/framing station.

4.3.3 Modification iterations.

After all of the subassemblies have been processed and built into a basic vehicle framework at the framing station(s), their makespan is matched with the available build time. If it matches, it is declared a feasible solution, and stored into a database of feasible solutions, but if not, the first iterative process of layout and contents modification begins. A feasible solution is defined as a layout capable of processing the required number of subassemblies in a shift, as well as being consistent with the constraints, including the budget, but not yet matching the objective, that is, to minimise the cost. However, the first layout is inevitably infeasible and SIMAID attempts to approach the feasibility zone by using the following rules: First, SIMAID checks:

- Which cell has had the biggest blocking effect (longest queues) on the subassemblies;
- Which process p has been in greatest demand throughout the simulation run.

SIMAID now enters the facility modification phase. The program will attempt to move from its currently infeasible solution to a more feasible one, i.e. approaching feasibility. It does this by noting the above requirements and adding to the present facility the corresponding processes and/or cells. The control system for this is outlined in fig. 4.8 below:

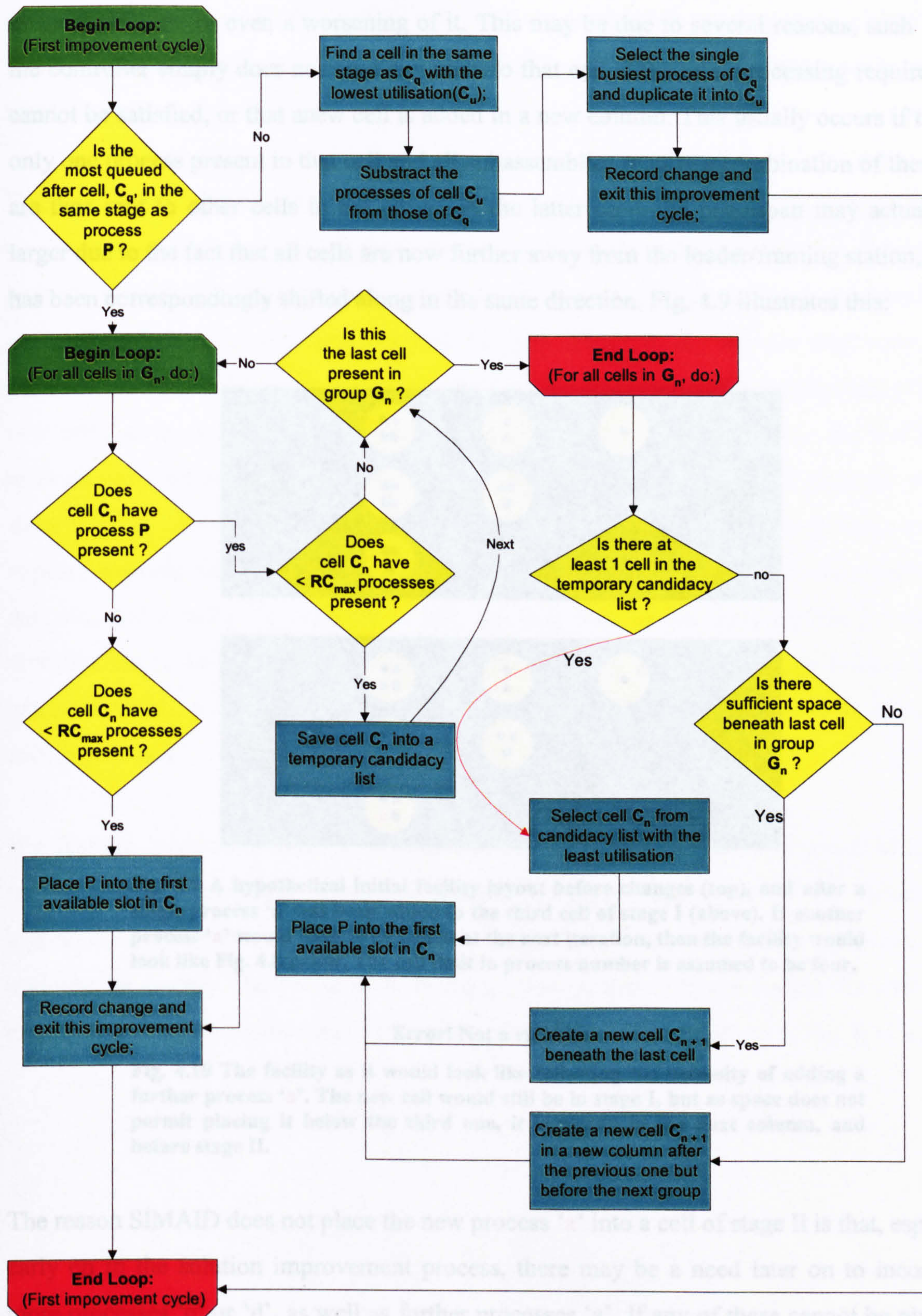


Fig 4.8 The first cycle of improvements iterations.

Once a process has been placed, the second simulation iteration begins, much like the first one. Upon its ending, the makespan is noted again and a decision is made based on the effect, if any, the recent change to the facility has made. It may be that there is no improvement at all

in the makespan, or even a worsening of it. This may be due to several reasons, such as that the controller simply does not send any units to that cell if the total processing requirements cannot be satisfied, or that a new cell is added in a new column. This usually occurs if there is only one process present in that cell and all subassemblies require a combination of them, and are thus sent to other cells in the group. In the latter case, the makespan may actually get larger due to the fact that all cells are now further away from the loader/framing station, which has been correspondingly shifted along in the same direction. Fig. 4.9 illustrates this:

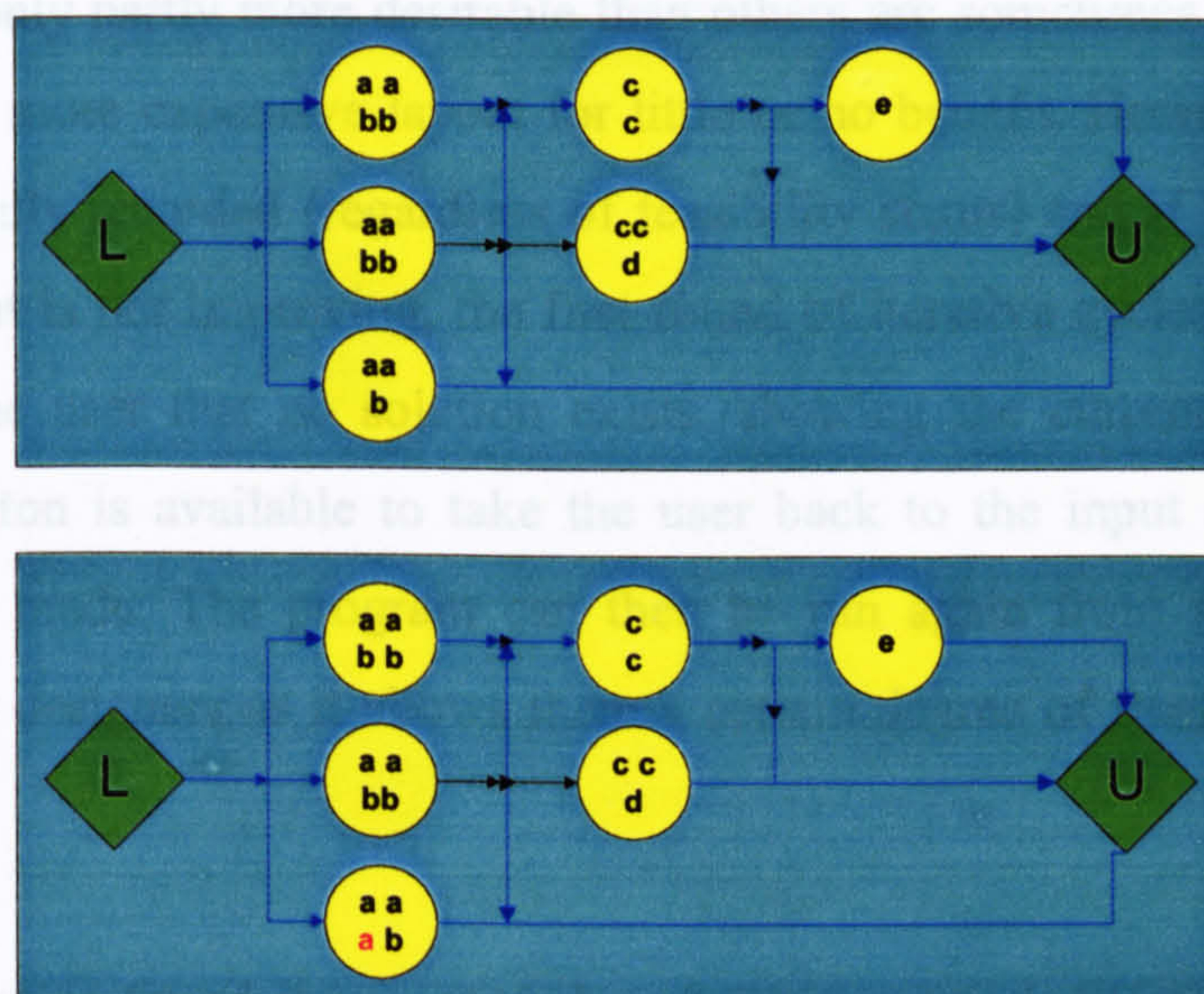


Fig. 4.9 A hypothetical initial facility layout before changes (top), and after a single process 'a' has been added to the third cell of stage I (above). If another process 'a' would have to be added at the next iteration, then the facility would look like Fig. 4.8 below. The cell limit in process number is assumed to be four.

Error! Not a valid link.

Fig. 4.10 The facility as it would look like following the necessity of adding a further process 'a'. The new cell would still be in stage I, but as space does not permit placing it below the third one, it is placed in the next column, and before stage II.

The reason SIMAID does not place the new process 'a' into a cell of stage II is that, especially early on in the solution improvement process, there may be a need later on to incorporate more processes 'c' or 'd', as well as further processes 'a'. If any of these cannot be placed in any of the cells of their own stage, then processing mis-combination may result, in that some subassemblies may require the use of a specific process combination which is negated by the new order. In this case, these subassemblies cannot be processed by the new layout and consequently it will be an infeasible one. Though not probable in the above example,

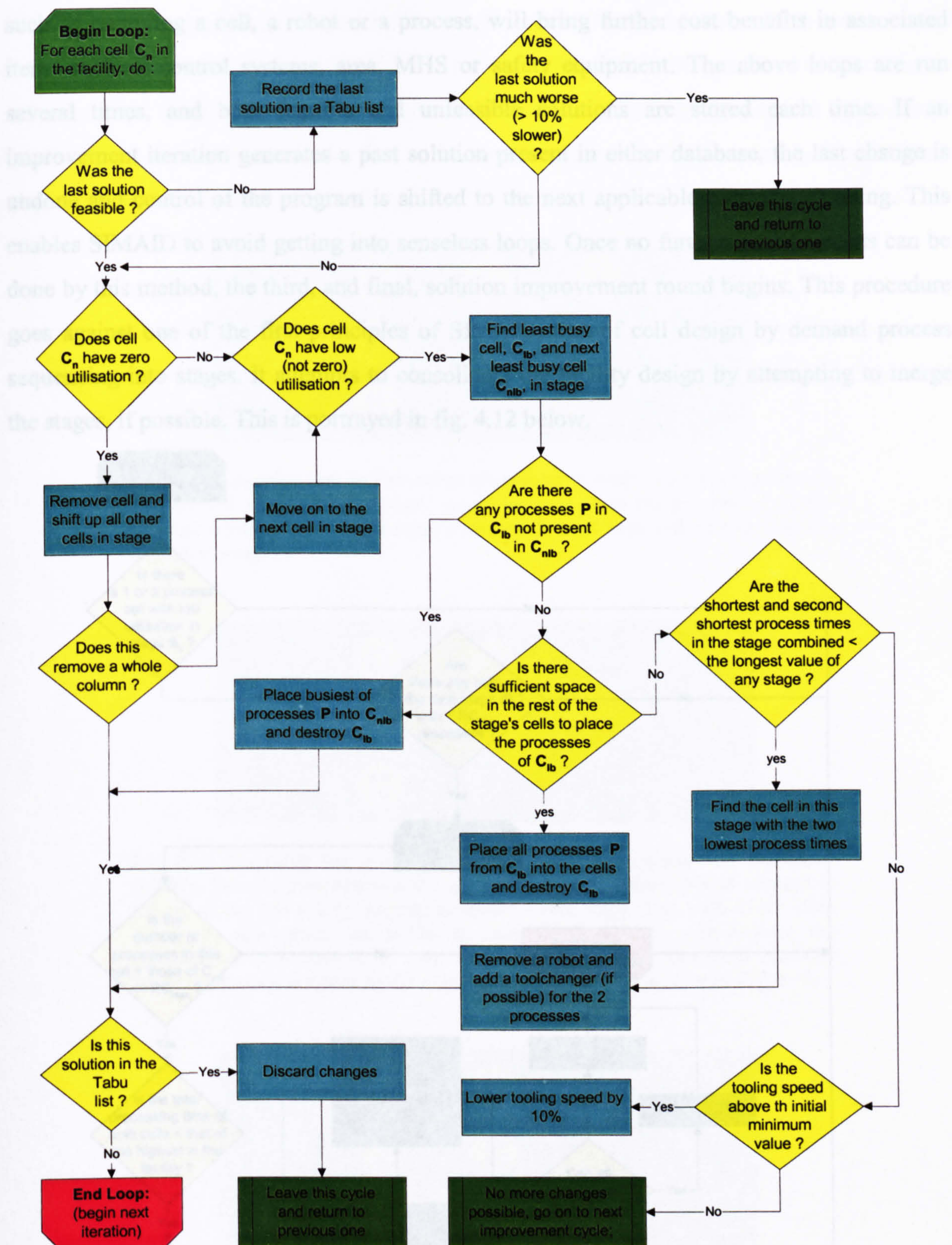


Fig 4.11 The second cycle of improvements iterations.

If all cells in the facility have non-zero and non-low utilisation levels, there is little optimisation that can be done, and further attempts to reduce costs will create either an infeasible solution or, at most, an equivalent one. It should be remembered that any change,

such as removing a cell, a robot or a process, will bring further cost benefits in associated items such as control systems, area, MHS or safety equipment. The above loops are run several times, and both feasible and unfeasible solutions are stored each time. If an improvement iteration generates a past solution present in either database, the last change is undone and control of the program is shifted to the next applicable part of the coding. This enables SIMAID to avoid getting into senseless loops. Once no further improvements can be done by this method, the third, and final, solution improvement round begins. This procedure goes against one of the first principles of SIMAID, that of cell design by demand process sequencing into stages. It attempts to consolidate the facility design by attempting to merge the stages, if possible. This is portrayed in fig. 4.12 below.

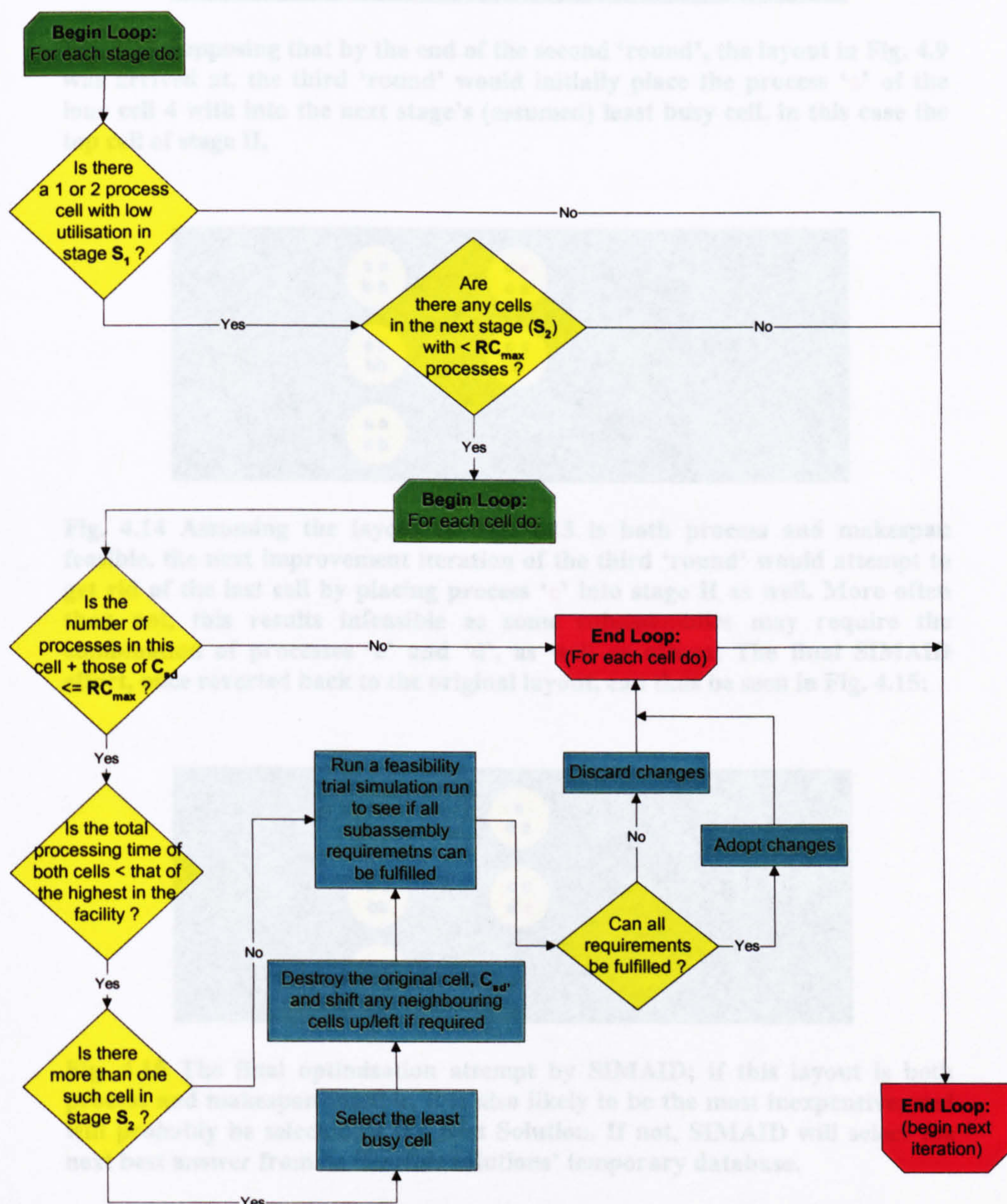


Fig 4.12 The third and final cycle of improvements iterations.

The above control code comprises the last optimisation attempt by SIMAID, but experience has shown this rarely works. Usually, by this stage there is little optimisation that can be achieved, and if any cells can be merged as described, this is usually negated by the non-feasibility of the resulting layout, due to incomplete subassembly processing requirements. Figs. 4.13 to 4.14 below shows an example of the above:

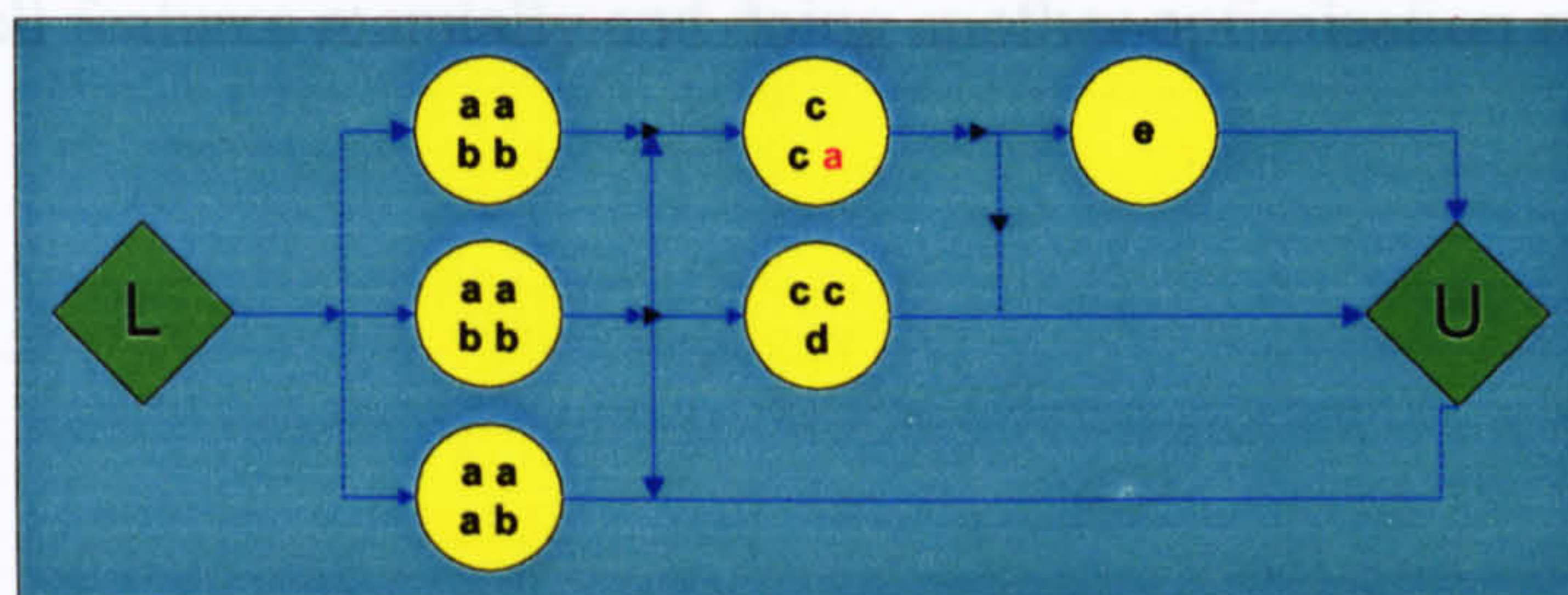


Fig. 4.13 Supposing that by the end of the second 'round', the layout in Fig. 4.9 was arrived at, the third 'round' would initially place the process 'a' of the lone cell 4 with into the next stage's (assumed) least busy cell, in this case the top cell of stage II.

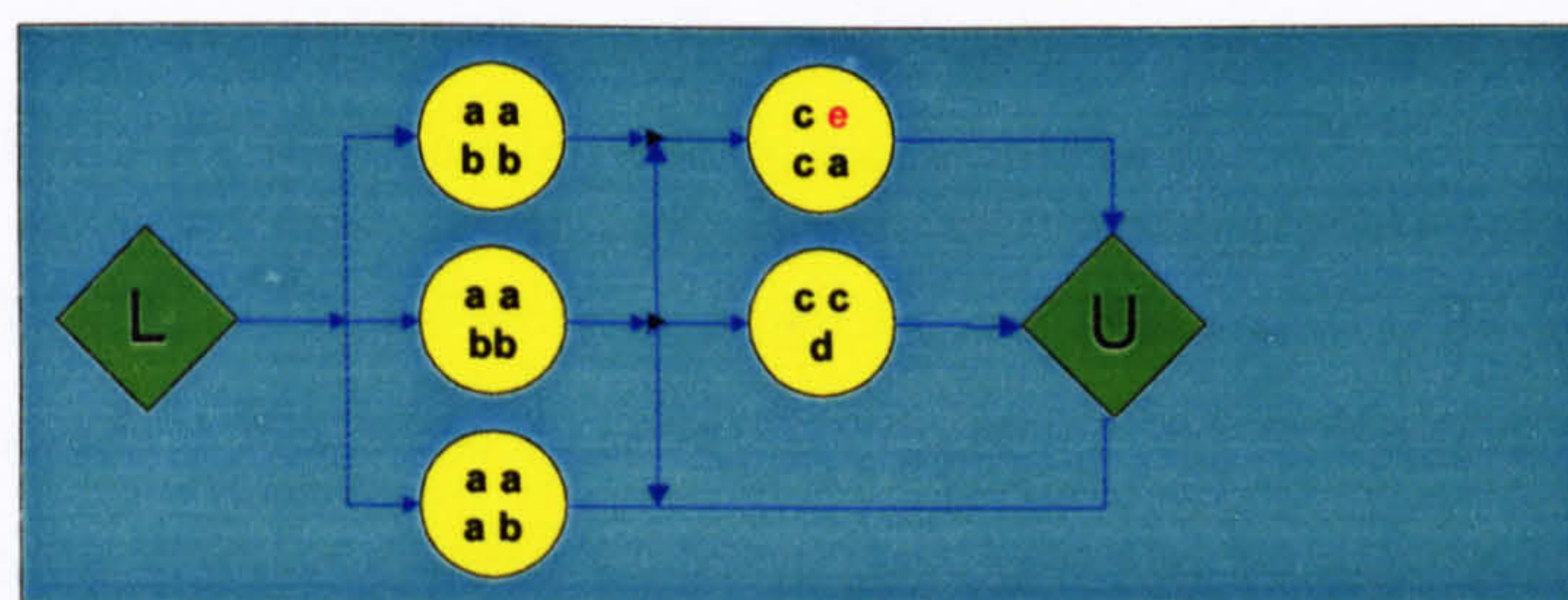


Fig. 4.14 Assuming the layout in Fig. 4.13 is both process and makespan feasible, the next improvement iteration of the third 'round' would attempt to get rid of the last cell by placing process 'e' into stage II as well. More often than not, this results infeasible as some subassemblies may require the combination of processes 'e' and 'd', as well as others. The final SIMAID effort, once reverted back to the original layout, can thus be seen in Fig. 4.15:

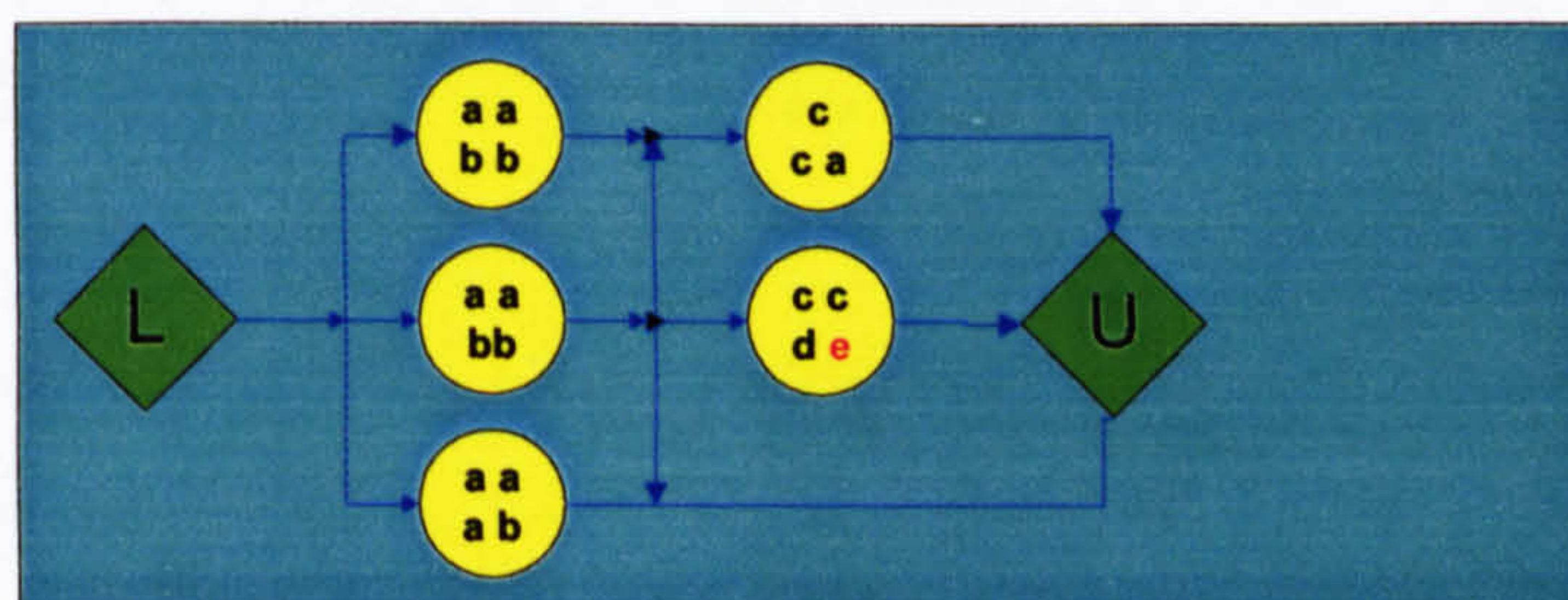


Fig. 4.15 The final optimisation attempt by SIMAID; if this layout is both process and makespan-feasible, it is also likely to be the most inexpensive, and will probably be selected as the Best Solution. If not, SIMAID will select the next best answer from its 'feasible solutions' temporary database.

Once no more improvements are possible the third iterative improvement loop stops and the user is presented with the best layout SIMAID could find. The user has a choice of either accepting this solution, or going back to the input screens to change an initial variable. Of course, if sensitivity analyses need to be done, this would be the normal way to do them anyway. Other options not yet incorporated could be the ability to create 'what if' scenarios, changing any end cell features manually and doing another optimisation run.

Subassembly	No. of parts	Load time
1	10	33
2	5	21
3	3	29
4	2	19
5	2	18
6	3	23

Table 5.1 Subassembly data inputted into SIMAID.

The subassembly loading time (time taken to load all of the components/parts onto the mobile tooling) varies between subassembly, but in SIMAID there is only provision for the input of a single value. This drawback will be addressed in due course, but when this example was run the ability to enter individual load times was not available. This problem was circumvented by taking an average of all load times, considering the number of parts involved. The final value entered into SIMAID was 7 seconds. The other details were:

- Available length: 38 metres;
- Available width: 14 metres;
- Available working time: 47 weeks a year, 5 days a week, 3 x 6.6-hour shifts;
- Volume, per year: 180,000;
- Assembly size: 1.4 x 1.6 metres;
- Number of subassemblies: 6 (10, 5, 3, 2, 2, 3 parts each);
- No. Processes: 2 (6 spot welding types, 1 bolting and 1 adhesive);
- Total number of spot welds: 364 at 2.8 seconds/spot, for subassemblies 1-3 and 5;
- Total adhesive application: 4 metres at 0.2 metres/second, for subassembly 4;
- Total number of bolts: 16 at 3.5 seconds per bolt, for subassembly 6;
- Budget: £ 2.5 million;

Chapter 5 – Case studies

5.1 Industrial example 1.

5.1.1 The data

This example derives from a real proposal to a major manufacturer by a first tier supplier, which is involved in line building. A single model subframe needed to be built within an existing factory. The number of subassemblies is 6, with the following characteristics:

Subassembly	No. of parts	Load time
1	10	33
2	5	21
3	3	20
4	2	18
5	2	18
6	3	22

Table 5.1 Subassembly data inputted into SIMAID.

The subassembly loading time (time taken to load all of the components/parts onto the mobile tooling) varies between subassembly, but in SIMAID there is only provision for the input of a single value. This drawback will be addressed in due course, but when this example was run the ability to enter individual load times was not available. This problem was circumvented by taking an average of all load times, considering the number of parts involved. The final value entered into SIMAID was 7 seconds. The other details were:

- Available length: 38 metres;
- Available width: 14 metres;
- Available working time: 47 weeks a year, 5 days a week, 3 x 6.6-hour shifts;
- Volume, per year: 180,000;
- Assembly size: 1.4 x 1.6 metres;
- Number of subassemblies: 6 (10,5,3,2,2,3 parts each);
- No. Processes: 8 (6 spot welding types, 1 bolting and 1 adhesive);
- Total number of spot welds: 364 at 2.8 seconds/spot, for subassemblies 1-3 and 5;
- Total adhesive application: 4 metres at 0.2 metres/second, for subassembly 4;
- Total number of bolts: 16 at 3.5 seconds per bolt, for subassembly 6;
- Budget: £ 2.5 million;

5.1.2 The results

The data above is incomplete but serves to indicate the problem type. SIMAID found its optimal solution at iteration 20, generating a makespan of 6.47 hours per shift. The layout is shown in figure 5.1.1 below:

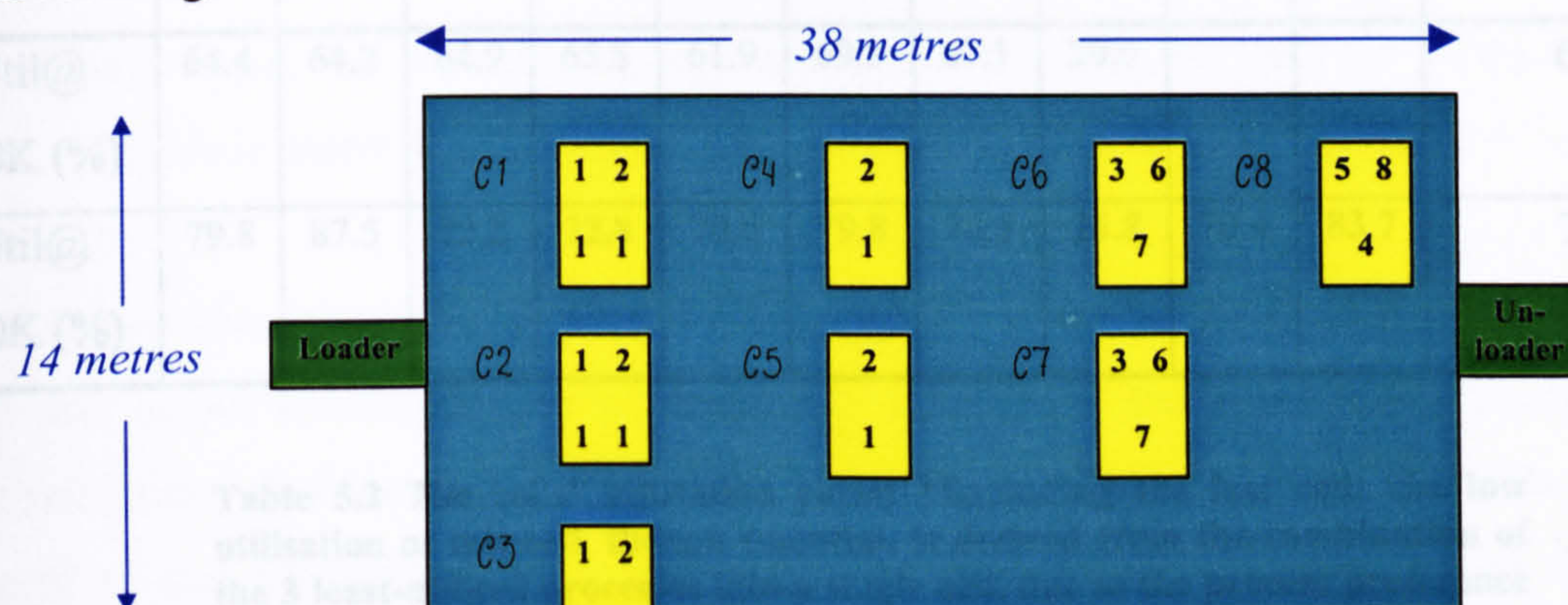


Fig 5.1 The facility layout generated by SIMAID (not to scale).

When the volume was increased to 240,000 units per annum, SIMAID interestingly added two cells (new cells 9 and 10) instead of placing the processes of cell 6 into those of cell 4 or 5. The new layout can be seen in Fig 5.1.2 below:

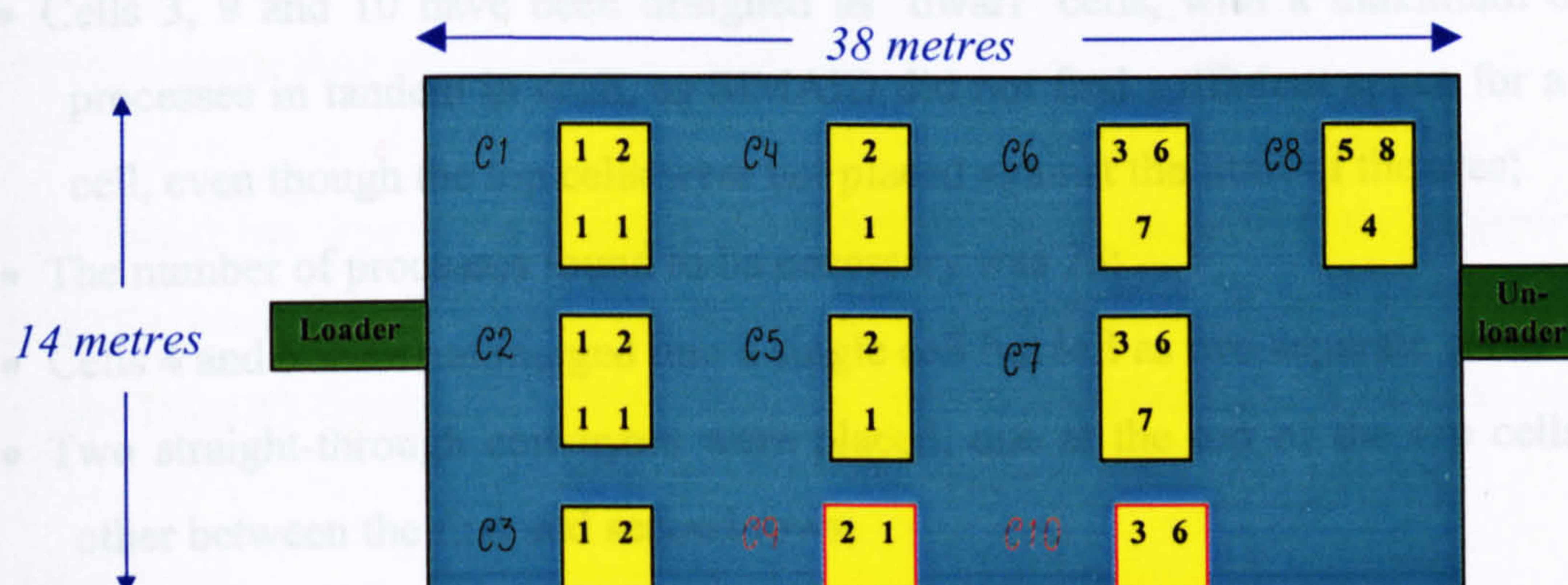


Fig 5.2 The facility layout generated at 240,000 units per annum.

The average utilisation also increased across the whole board. Within SIMAID, cell utilisation is taken as the percentage of time a cell is in use over total time. The time is taken from the start of a simulation, when the first set of parts are loaded onto the tooling, to when the last subassembly is unloaded from the tooling. Obviously this means that for part of the time (especially at the initial and final stages), until there are sufficient units in the system, most of the cells are not being utilised. A cell is defined as being in use when any of its processes are actually working on the unit, regardless of how long the latter actually stays within the cell itself. This has been designed as such due to the frequent occurrence of units having to remain

within cells, as their next target cell is busy at the time, especially if no buffers are present. Table 5.2 shows the results, below.

Cell:	1	2	3	4	5	6	7	8	9	10	Average util.*:
Util@ 180K (%)	64.4	64.2	64.9	65.8	61.9	69.8	67.1	20.6			65.4
Util@ 240K (%)	79.8	87.5	72.2	72.8	70.6	79.8	78.5	25.8	70.4	83.7	77.2

Table 5.2 The cells’ utilisation rates; *Excluding the last cell; the low utilisation of this cell, though essential, is derived from the combination of the 3 least-utilised processes into a single cell, due to the process preference indications;

5.1.3 Critical analysis

Several points are worthy of note:

- Cells 3, 9 and 10 have been designed as ‘dwarf’ cells, with a maximum of only 2 processes in tandem in each, as SIMAID did not find sufficient space for a full-size cell, even though the top cells were not placed against the limit of the area;
- The number of processes found to be necessary was 23;
- Cells 4 and 5 were not merged into a single cell but left as two separate cells;
- Two straight-through conveyors were placed, one at the top of the top cells and the other between the first and second rows;

A comparison with the company-designed and optimised layout reveals several interesting differences and similarities. The actual design (as well as the input data) can be seen in Appendix C1. It is worth noting that the company designed their version as a conventional ‘line’ layout, not as an agile facility.

- The layout is in the form of a line, with no ‘cells’ present as such, but the initial part has several workstations which have been designed as independent operating areas, in effect equivalent to cells;
- The number of robots found to be necessary was 22, one less than SIMAID’s;

- One robot (robot 05) is not used as processing robot but as transfer device instead; this brings the total number of process-bearing robots to 21, two less than SIMAID's.
- A single, straight main line conveyor and two, short side conveyors were used, but also an overhead hoist as well as three manual operators;
- Both loader and unloader stations are included within the 38 x 14 metre area, while SIMAID placed them outside, indicating a less efficient use of space.

5.1.4 Assumptions and limitations

Any use of an industrial case is invariably limited in terms of the assumptions and approximations that are required to be made, and this is no exception. The data had to be adapted so that SIMAID could accept it, and the approximations used (such as the loading time's collective averaging) worsen any direct comparison further. Two other factors affecting the exercise are the different design philosophy (linear versus cellular), and SIMAID's built-in limitations (such as the maximum number of processes allowed, etc.). Additionally, this design was generated without two important SIMAID attributes: No scheduling was done, and the third 'round' of solution improvement ('Stage merging') was not applied either, as neither part of the program is as yet functional.

SIMAID found the cost at just under £2,500,000, which is within the objective value. However, the costing system relies totally on the individual costs placed within its database, which were ballpark figures. For a more 'serious' exercise, accurate figures would be essential.

5.1.5 Discussion

SIMAID has successfully managed to design an agile facility for a current industrial example, designing a cellular layout with a number of processes, which approximates the figure devised by the company. The 180,000 units can easily be built in such a facility, and as shown in the 240,000-unit trial, with a generous degree of volume flexibility. However, several criticisms could be levelled at the program, which could form the basis of further improvements, should any of the industrial companies in the SALVO 6 project wish to take up the challenge.

1. Firstly, the result was not as close as desired (in terms of space taken, cell utilisation and processes used) due to the already mentioned facts of lack of scheduling and 'Stage merging'. A good schedule would have allowed SIMAID to reach feasible solutions faster,

without the need to add so many processes, and the use of stage merging may have removed an excess cell, lowering the total facility cost further.

2. Secondly, the use of space was poor. The loader/unloader were placed outside because SIMAID did not find sufficient space inside the designated area for all of the cells, but again, with fewer cells, as described above, this problem would probably be eliminated. However, this does not affect the fact that SIMAID incorporated two MHS 'corridors' for the subassemblies to run between the loader and the third stage cell (cell 8), which is unnecessary. It also has a knock-on effect on the shape and size of the cells at the bottom (cells 3, 9, and 10), which have only got two processes each but could hold up to 4 if the top cells were placed against the limits of the area. Clearly, a more intelligent space-optimising method is required.
3. The first stage's average utilisation was 64.24 %, which means that these cells were unutilised during approximately 35% of the time. Since the values are similar for all the cells, it means that each cell received only 20% of the total visits by the units to stage one. If the above problems were rectified, and a good schedule were in place minimising unit wait and blocking times, a whole cell in the first stage could easily be made to disappear by incorporating the processes into another cell of the same stage, or perhaps by deleting them entirely. Together with the better use of space, the whole second column of stage one would not be designed, allowing space for the loader/unloader and a faster makespan due to the closer proximity of stage two and three cells.
4. Finally, in such a limited space buffers could not have easily been incorporated, and in SIMAID this option is not as yet functional. However, it would have been interesting to see, if it were possible, the effect of placing a buffer say, in between the stages, would have on the makespan and cell efficiency.

5.2 Industrial example 2.

This exercise was also derived from a real-world example, the advantage here being the use of far more accurate costs as these were derived directly from the data source (the company). The parts to be assembled were the front doors of a small hatchback made in. A budget of 5,077,811 Euros was given, and at the then current rate of 1.4450 Euro to the £ this made it approximately £ 3,500,000. There are two models to be assembled, the 226LH/RH and the 256LH/RH (left/right hand side). Each model has two subassemblies, an interior and an exterior side. Each subassembly requires three joints (or focal points of operation), which in reality correspond to parts. The other relevant data is described below.

5.2.1 The data

- Available length (space): 25 metres;
- Available width (space): 21 metres;
- Available working time: 47 weeks a year, 5 days a week, 1107 minutes/day.
- Available shifts: 3 shifts, 6 hours and 15 minutes per shift;
- Volume, per year: 300,000 each (600,000 in total);
- Assembly size: 2 x 1.3 metres;
- No. subassemblies: 4 for each model: 2 -left & 2-right hand side, one interior and one exterior (= in 8 total);
- No. Processes: 8 (4 spot welding types, 1 adhesive, 1 induction heating, 1 sealant application and 1 hemming process);
- Total no. of weld spots: 35 (multi-welded) at 2.5 seconds/spot (all units);
- Assembly size: 570 x 630 x 950 mm (door-shaped);
- Total adhesive application: 2.15 metres at 0.2 metres/second, for joint 1 of the exterior parts of each subassembly;
- Budget: ~5M Euros (~ £ 3.5 million);

5.2.2 Data translation and incorporation

As this data has to be adapted to be used within SIMAID, several assumptions are made. There are 35 spotwelds to be done of four different types, but types I and II are multiwelders of 4 and 10 welds each, used on all the assemblies universally. If robots had to be used these spots would obviously be done one at a time, handicapping the facility's production rate but

enhancing its agility. Additionally, spotwelds of type III are also different for the two models, in effect making 5 different spot types. There are also 4 other processes that are required, meaning that the total number of processes to be used becomes 9. Since SIMAID can only handle a maximum of 8, some adjustments are required. Table 5.3 below illustrates this and table 5.4 shows the data translation used for the exercise:

	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>
Process	Multi-weld 4 spots	Multi-weld 10 spots	Weld 8 spots	Hemming adhesive application	Servo Hemmer	Weld 13 spots	Cosmetic sealer application	Induction curing

Table 5.3 Original process data.

	<i>P1</i>	<i>P2</i>	<i>P3</i>	<i>P4</i>	<i>P5</i>	<i>P6</i>	<i>P7</i>	<i>P8</i>
Process	Multi-weld 14 spots (from 1 & 2)	Multi-weld 8 spots for model 226	Multi-weld 8 spots for model 256	Weld 13 spots	Hemming adhesive application	Induction curing	Cosmetic sealer application	Servo Hemmer

Table 5.4 Translated process data.

Welding processes I and II were aggregated into P1, as these applied to all subassemblies anyway. The original process 3 was split into P2 and P3, one for each model, as in reality multiwelders are difficult to adapt to different models. Process 5 (servo-hemming) is migrated to P8, process 6 to P4, process 7 stays the same and process 8 becomes P6. The actual process naming order (P1, P2, ... etc.) is irrelevant, as the original process sequence is retained.

Appendix C2 gives the actual data input. As can be seen, every subassembly uses process 1, and the number of welds is split to make up the total of 14 per subassembly. The same applies to processes P2 (used by model 226) and P3 (used by model 256). The 13 spots of process P4 are also split into two for the third joint of each subassembly. Processes 5,6,7 and 8 apply once for each subassembly, and therefore are uniformly distributed across the range. The values for processes 5 (adhesive application) and 7 (cosmetic sealer application) are explained below. The process preferences are devised according to the original data sequence. The actual window can be seen in Fig. 5.3 below:

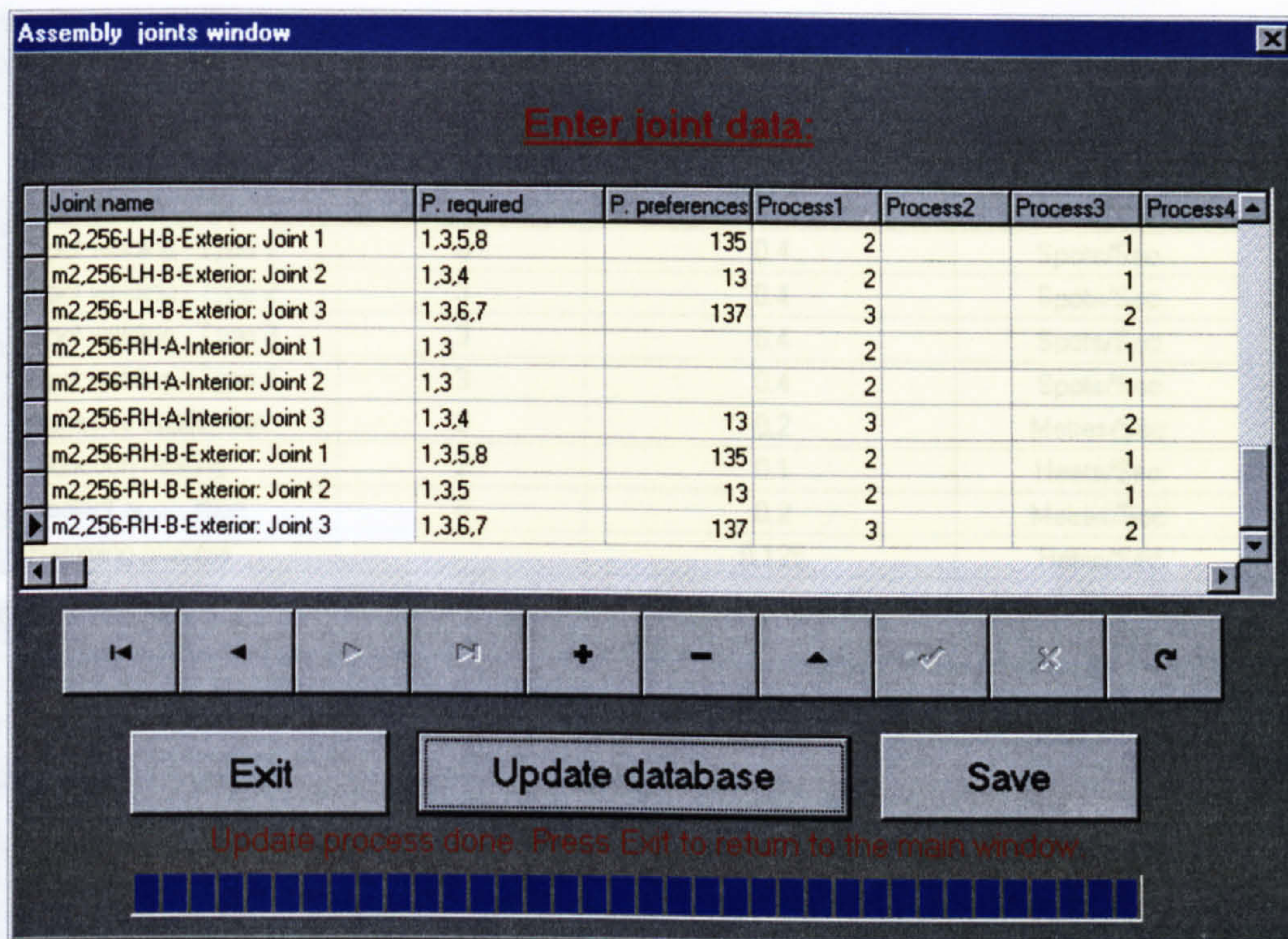


Fig. 5.3 The actual SIMAID data window.

The process rates of work are also company values, derived from real industry data. Table 5.5 below gives the original data:

	<i>P1</i>	<i>P2</i>	<i>P3</i>	<i>P4</i>	<i>P5</i>	<i>P6</i>	<i>P7</i>	<i>P8</i>
Processes	2.5 seconds per weld	2.5 seconds per weld	2.5 seconds per weld	2.5 seconds per weld	Adhesive: 5 seconds per metre	Induction heating: 10 secs. (total)	Sealant: 5 seconds per metre	Hemming: 8 seconds (total)

Table 5.5 Translated process data.

All process rates have to be converted as units per second. For the welding processes, this translates 2.5 seconds of weld time as 0.4 welds per second. Processes 5 and 7 apply the liquids to the door external dimensions, ($570 + 630 + 950 = 2150$ mm) at a rate of 0.2 metres per second. As the values are all in metres, this means that these two processes are required in 2.15 'units' for each subassembly. Finally, the '*Estimated time between same-process operations on the same joint*' value, corresponding to the time a robot arm takes to move between joints of the same subassembly, is a realistic 2 seconds¹. Fig 5.4 below shows the actual process selection window used in SIMAID:

¹ Personal communication from a Lamb Technicon production engineer.

Process selection window

Enter process type and attributes into database:

Process name	Tool weight (Kg)	Tool speed (Units/time)	Units
Spot welding - Type 1	3	0.4	Spots/Sec
Spot welding - Type 2	3	0.4	Spots/Sec
Spot welding - Type 3	3	0.4	Spots/Sec
Spot welding - Type 4	3	0.4	Spots/Sec
Adhesive application	3	0.2	Metres/Sec
Induction heating	2	0.1	Heats/Sec
Sealant application	5	0.2	Metres/Sec
Hemming process		0.125	Hems/Sec

Navigation buttons:

Enter estimated time between same-process operations on the same joint :

Exit **Done**

Fig. 5.4 The process selection window.

Cost data inputted reflected both the actual vales given for equipment and the assumptions made due to the different facility design principles. As SIMAID has been designed, from initial concepts, as a tool for designing cell-based agile production facilities, incorporating concepts such as moveable assembly tooling and reconfigurable cells, some fairly large assumptions have to be made. The original design includes conveyors and buffers, which have not been incorporated into SIMAID. Those costs have therefore been ignored. However, additional costs included have been:











- The moveable tooling (at a £60,000 per unit cost), based on work conducted at the SALVO cell at the Warwick Manufacturing Group.
- Cell ancillaries and control system values reflect the cell-based methodology, as these would not be applicable to conventional designs.
- The values for buffers, the framing station and toolchangers are not applicable so are not used.
- Finally, P6 and P8 robot values were zero as these processes, being entirely self-contained and static, automated machines cannot possibly use robots.

Fig. 5.5 shows the data entered into SIMAID:

Costs data input

Enter cell and facility cost data:

Item to cost	1	10	100
Buffer, single unit			
Buffer, variable, per unit			
Cell ancillaries	15		
Control system, per cell	5		
Framing station			
P1 robot	25		
P2 robot	25		
P3 robot	25		
P4 robot	25		
P5 robot	25		
P6 robot	0		
P7 robot	25		
P8 robot	0		
Proc1	288		
Proc2	106		
Proc3	106		
Proc4	55		
Proc5	287		
Proc6	340		
Proc7	287		
Proc8	1095		
Site	598		
Toolchanger, 2-process	0		
Tooling	40		

Navigation buttons:          

Save changes and exit

Exit without saving changes

Fig. 5.5 Cost data used for the exercise (in Euros).
Appendix C2 shows the data as supplied by the company.

The constraints used have been taken from the data sheet listed in section 5.2.1 and include the budget, the available space and the production time window. The values are in £s instead of Euros as all costs are translated into £. The 6 hours and 15 minutes per shift is derived from the 1107 available minutes per day. The constraints data window can be seen below.

Constraints data input

Enter constraint data to limit search range:

Maximum facility cost (£): £: 3500000

Maximum facility Length (in metres): 25 metres

Maximum facility Width (in metres): 21 metres

Minimum number of routing alternatives: 0

Maximum number of robots per cell: 4

Maximum number of shifts per 24 hours: 3

There are 6 hours and 15 minutes in each shift.

There are 47 working weeks a year of 5 days each.

Cell data Done Cancel

Fig. 5.6 The constraints window.

5.2.3 The result

Finally, the cell data is inputted. The assembly size is 950 mm at its largest, hence the 1 metre value. The Material Handling System (MHS) width is taken to be twice this, considering the tooling and safety measures. Component loading times are between 3 and 6 seconds, 5 being the value used here. Tooling travel velocity between the cells is limited to between 1 and 4 metres per second, as faster speeds become unrealistic and/or expensive (in terms of equipment that would need to be incorporated). The MHS organisation is side-by-side (in parallel) as the vertical option gives unpredictable results. Finally, toolchanger use is disabled, as this has not yet been implemented. Fig 5.7 below shows the cell data window with the input information.

- There are four each of processes 1 and 2, two processes 3 and 4, and one each of the rest;
- Even though space in existing cells was available, SIMAID did not place the additional processes 1, 3 or 4 within them, preferring to put them in cells of their own;
- Utilisation is high for the first two cells only, falling off to virtually (but not quite) zero for cells 3, 5, and 7;
- Utilisation for cells 4 and 6 is roughly half that of cells 1 and 2;

Cell data window

Enter the following data:

Material Handling System width (in metres):

Average component loading time (secs):

Min tooling transport system's speed(m/s):

Max tooling transport system's speed(m/s):

Assembly size - length (in metres)

Assembly size - width (in metres)

☒ MHS : Directions placed in parallel
☐ MHS : Directions placed vertically

Toolchanger information

☐ Disallow toolchanger use, 1 process/robot and/or 1 robot/process only;
☐ Allow 4-tool (2-robot, 2-process) toolchangers only;
☐ Allow multiple process toolchangers in underutilised cells only;
☒ Allow multiple process toolchangers in any cell (let me do the thinking).

☒ OK

Fig. 5.7 The cell data window.

5.2.3 The result

SIMAID's best answer is an apparently infeasible solution at the 18th iteration, generating a makespan of 268 seconds for 40 subassemblies, or 6 hours and 24 minutes (9 minutes over target time). Fig. 5.2.6 shows the results. Its unfeasibility stems from the extra time required to make the parts, even though the facility cost was well within the budget. Cell utilisations were as follows:

Cell:	1	2	3	4	5	6	7	Average util.
Utilisation (%)	78.2	73.9	2.9	35.8	1.3	23.8	7.14	31.86

Table 5.6 The cell utilisation levels.

Several points should be noted:

- There are four each of processes 1 and 2, two processes 3 and 4, and one each of the rest;
- Even though space in existing cells was available, SIMAID did not place the additional processes 1, 3 or 4 within them, preferring to put them in cells of their own;
- Utilisation is high for the first two cells only, tailing off to virtually (but not quite) zero for cells 3, 5, and 7;
- Utilisation for cells 4 and 6 is roughly half that of cells 1 and 2;

- The loader is horizontally half-way between cells 2 and 3, while the unloader is between the two last cells;
- The cost is £140,000 below budget.

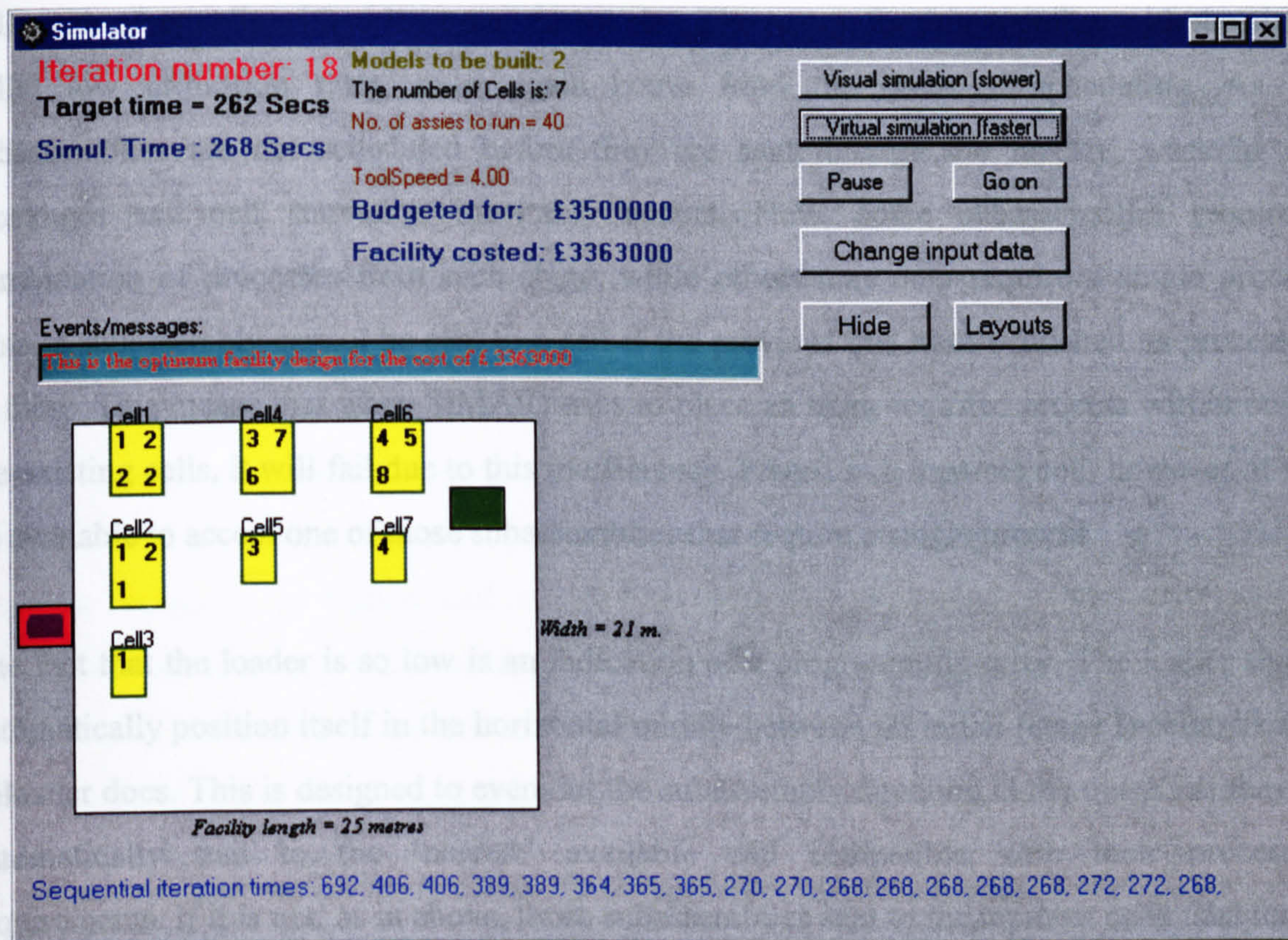


Fig. 5.8 The results window.

5.2.4 Critical analysis and discussion

The actual process times required by the different processes (from P1 to P8) in order to process all of the required units (subassemblies) are approximately 12:6:6:5:1:1:1:1. In effect, there should be 12 times as many processes P1s as P5s, P6s, etc. But SIMAID only quadrupled the amount of P1s and P2s in terms of actual processes within the cells. When seen in conjunction with cell utilisations, however, it becomes clearer why this was sufficient. For the given volume, SIMAID found that 4 processes each were sufficient to enable the makespan. Proportionally, this would mean that the number of processes 5-8 should be 1/3. You cannot have 1/3 of a process present in a facility, but you can have one and use it for 1/3 of the time ... hence the low utilisation rates of the later cells.

SIMAID's first attempt is to generate a first layout based on the proportional process requirements, incrementing the number of processes in cells pro-rata as required. It first tries to place processes in the same stage required by the same subassemblies together, and if this fails it creates additional cells to put them in. In all three stages, there are single-processed cells with other cells with at least one vacant slot. The reason for this, together with the single cells' low utilisation rates, must again come from the issue of scheduling. As the subassemblies are not scheduled before they are sent through the facility, wasteful cell blockages and cell starvation inevitably occurs. Now, some subassemblies require a combination of processes from each stage, while others may only require a single process. Also, a subassembly cannot be sent to a cell if the previous one hasn't finished its processing in there. This means that when SIMAID tries to place an extra-required process within one of the existing cells, it will fail due to this inefficiency. Placed in a separate cell, however, it will be available to accept one of those subassemblies that require a single process.

The fact that the loader is so low is an indication of a programming error. The loader should automatically position itself in the horizontal middle between all initial (stage I) cells, like the unloader does. This is designed to even out the subassembly direction at the outset, as they are automatically sent to the 'nearest' available cell compatible with their processing requirements. If it is not, as in above, those subassemblies sent to the topmost cells take longer than they should, and over the whole simulation period this could perhaps help explain the 6 seconds' makespan excess that made this design 'infeasible'. If compared to the scheduling issue, however, this should prove to be of minor importance.

The cost found, at £140,000 under budget indicates that, though every effort was made, not all costs were placed into the database for consideration by SIMAID. Perhaps some of the costs were also misleadingly applied, such as the tooling entry; only seven tools (one per cell) were calculated, but in reality this could be higher as we do not know how many such tools are necessary to move all of the subassemblies around the cells at one time. The costs of 'cell ancillaries' and 'control systems' are also ballpark figures; these may be higher or lower, generating some uncertainty in the equation.

Comparison of the SIMAID-generated layout with the company-designed facility put SIMAID style drawing in here for comparison (seen in Appendix C2) is interesting. SIMAID calculated

the number of necessary robots as 16, twice the company-designed number. However, the two are not directly comparable. Due to the cellular layout, SIMAID attaches one robot to each process; the company has multiwelders and hemming machines in place, together with a curing oven. If we consider a machine a process, then the latter two would match the original design (1 each). However, comparing two multiwelders for 14 spots with four single-process robots is a little more difficult. An explanation could be in the fact that the original design includes *four* separate workstations, two on each side for each model. While the amount of work to be done is equivalent, time and scheduling constraints mean that you could not, perhaps, put just 2 of them into two cells and expect a 100% utilisation.

Explaining the four processes P2s is a little trickier. There should only be two of them, according to the process times requirement ratios mentioned above, yet SIMAID created four. Probably the best explanation is the implementation of the SIMAID program. While SIMAID searches for either the busiest cell *or* the most requested processes, it obviously does not distinguish when to use which system correctly in all occasions. A better-written program would enable the search to be guided towards the most needed requirement *all* of the time. Had it done so in the above example, it would have found that two processes (perhaps one in each cell) would have been sufficient.

As a final point, in comparison with the original facility design, the SIMAID-generated layout does not provide for buffers; the cells themselves act as buffers. In the company-designed facility, the engineers have allowed for a 3-unit buffer between Workstations 2 and 4 on each side (six units' worth), thus considerably easing the pressure on the cells and greatly reducing the chances of blockage or starvation.

5.3 Industrial example 3

The project involves the assembly of several components into the front structure of a product of a major manufacturer. Several subassemblies have to be assembled forming a larger final subassembly approximating to one quarter of the vehicle's BIW structure. This example took two zones, the dash assembly zone and the front structure assembly plus respot zone.

5.3.1 The data

• Available area	$300 \text{ m}^2 + 700 \text{ m}^2 = 1000 \text{ m}^2$
• Available length (space):	32 metres;
• Available width (space):	31 metres;
• Available working time:	46 weeks a year, 5 days a week, 908 minutes/day.
• Available shifts:	2 shifts, 7 hours and 34 minutes per shift;
• Volume, per year:	300,000 units;
• Assembly size:	1.3 x 1.3 metres;
• No. subassemblies:	2 (Front structure and panel dash);
• No. of joints:	5 (3 for the panel dash, 4 for the front structure);
• No. Processes:	8 (3 spot welding types, 2 stud welding, 1 projection welding, and 1 sealing processes);
• Total no. of spot welds:	238 at 2.5 seconds/spot (all units); 76 in the front structure, 50 for the dash panel, and 112 for the combined assembly;
• Total no. of projection welds:	11 at 2.5 seconds/spot;
• Total no. of stud welds:	13 at 2.5 seconds/spot;
• Assembly size:	1290 mm x 1280;
• Total sealant application:	4.2 metres at 0.2 metres/second;
• Budget:	£ 5,516,455 million;

5.3.2 Data translation and incorporation

The two subassemblies require different processes and need to be married into a final assembly forming the front structure, as shown in Fig. 5.3.1 below.

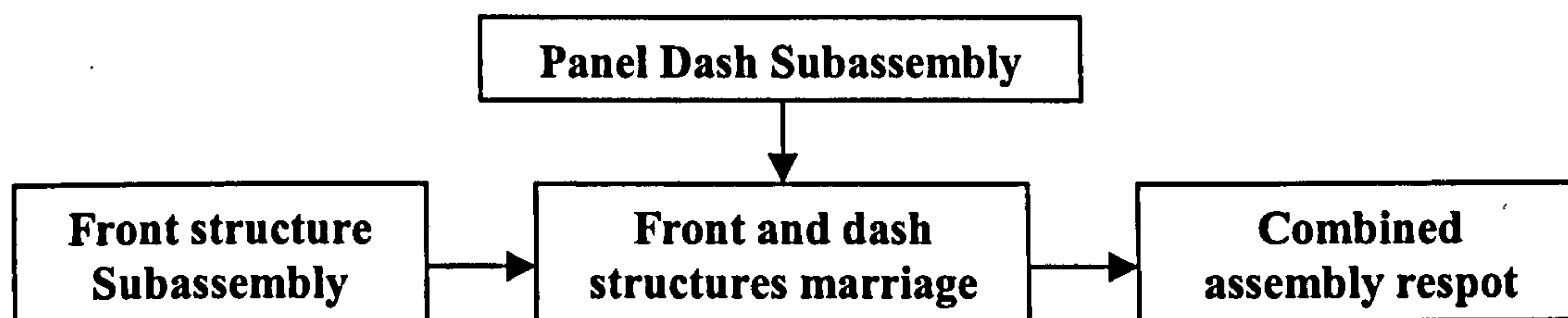


Fig. 5.9 The overall process sequence

The processes used are shown in table 5.7 below:

Process no.:	Description:	Rate (per second):
1	Front structure assembly spotwelding	0.4
2	Panel dash assembly bolting	1
3	Panel dash assembly studwelding, type 1	0.4
4	Panel dash assembly sealant application	0.2
5	Panel dash assembly projection welding	0.4
6	Panel dash assembly spotwelding	0.4
7	Subassembly marriage cell	0.3
8	Combined assembly final spotwelding	0.4

Table 5.7 The processes required.

There is also a strict process sequence to be observed. Table 5.8 below shows the individual subassembly actual process requirements:

Joints/Process:	1	2	3	4	5	6	7	8
Dash Panel, Joint 1		First		Second				
Dash Panel, Joint 2		First			Second			
Dash Panel, Joint 3			First			Second		
Front structure, Joint 1	First						Second	Third
Front structure, Joint 2	First							Second
Front structure, Joint 3	First							Second
Front structure, Joint 4	First							Second

Table 5.8 Original process data for the front structure assembly zone, showing the order of processing for the individual joints of each subassembly. As process 7 is the marriage station, and all of the parts are already on the mobile tooling, only one such processing requirement is necessary for the whole subassembly.

5.3.3 The results

As with the other examples, the number of units selected for the simulation was 20 (10 front structure and 10 panel dash). As simulating a whole shift would take far too long, a production

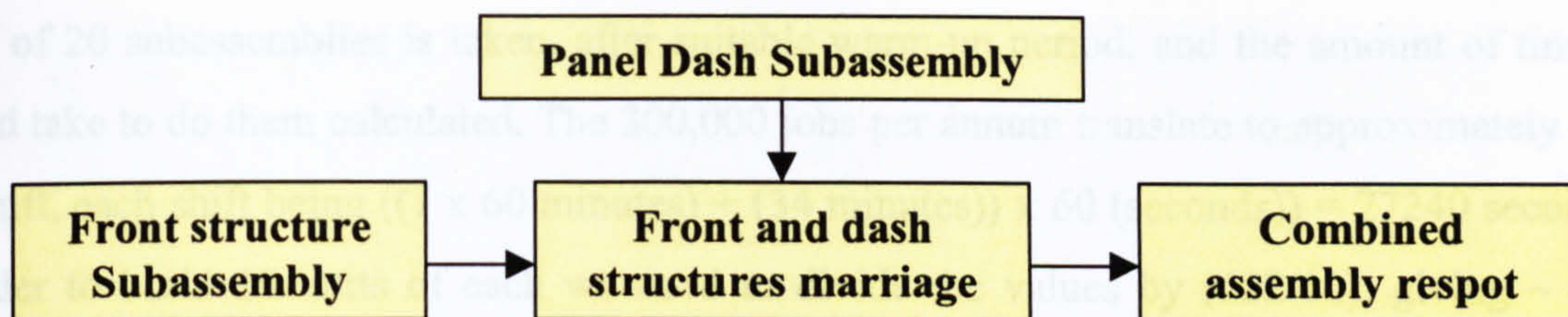


Fig. 5.9 The overall process sequence

The processes used are shown in table 5.7 below:

Process no.:	Description:	Rate (per second):
1	Front structure assembly spotwelding	0.4
2	Panel dash assembly bolting	1
3	Panel dash assembly studwelding, type 1	0.4
4	Panel dash assembly sealant application	0.2
5	Panel dash assembly projection welding	0.4
6	Panel dash assembly spotwelding	0.4
7	Subassembly marriage cell	0.3
8	Combined assembly final spotwelding	0.4

Table 5.7 The processes required.

There is also a strict process sequence to be observed. Table 5.8 below shows the individual subassembly actual process requirements:

Joints/Process:	1	2	3	4	5	6	7	8
Dash Panel, Joint 1		First		Second				
Dash Panel, Joint 2		First			Second			
Dash Panel, Joint 3			First			Second		
Front structure, Joint 1	First						Second	Third
Front structure, Joint 2	First							Second
Front structure, Joint 3	First							Second
Front structure, Joint 4	First							Second

Table 5.8 Original process data for the front structure assembly zone, showing the order of processing for the individual joints of each subassembly. As process 7 is the marriage station, and all of the parts are already on the mobile tooling, only one such processing requirement is necessary for the whole subassembly.

5.3.3 The results

As with the other examples, the number of units selected for the simulation was 20 (10 front structure and 10 panel dash). As simulating a whole shift would take far too long, a production

'slice' of 20 subassemblies is taken, after suitable warm-up period, and the amount of time it should take to do them calculated. The 300,000 jobs per annum translate to approximately 652 per shift, each shift being $((7 \times 60 \text{ minutes}) + (34 \text{ minutes})) \times 60 \text{ (seconds)} = 27240 \text{ seconds}$. In order to build 10 units of each we need to divide the values by $(652/10)$, giving ~ 413 seconds of run time. This then becomes the 'target' time for SIMAID, the maximum amount of time, after the warm-up period, it should allow in order to be able to make the 20 (proportionally 1 shift's worth) of subassemblies. Fig 5.10

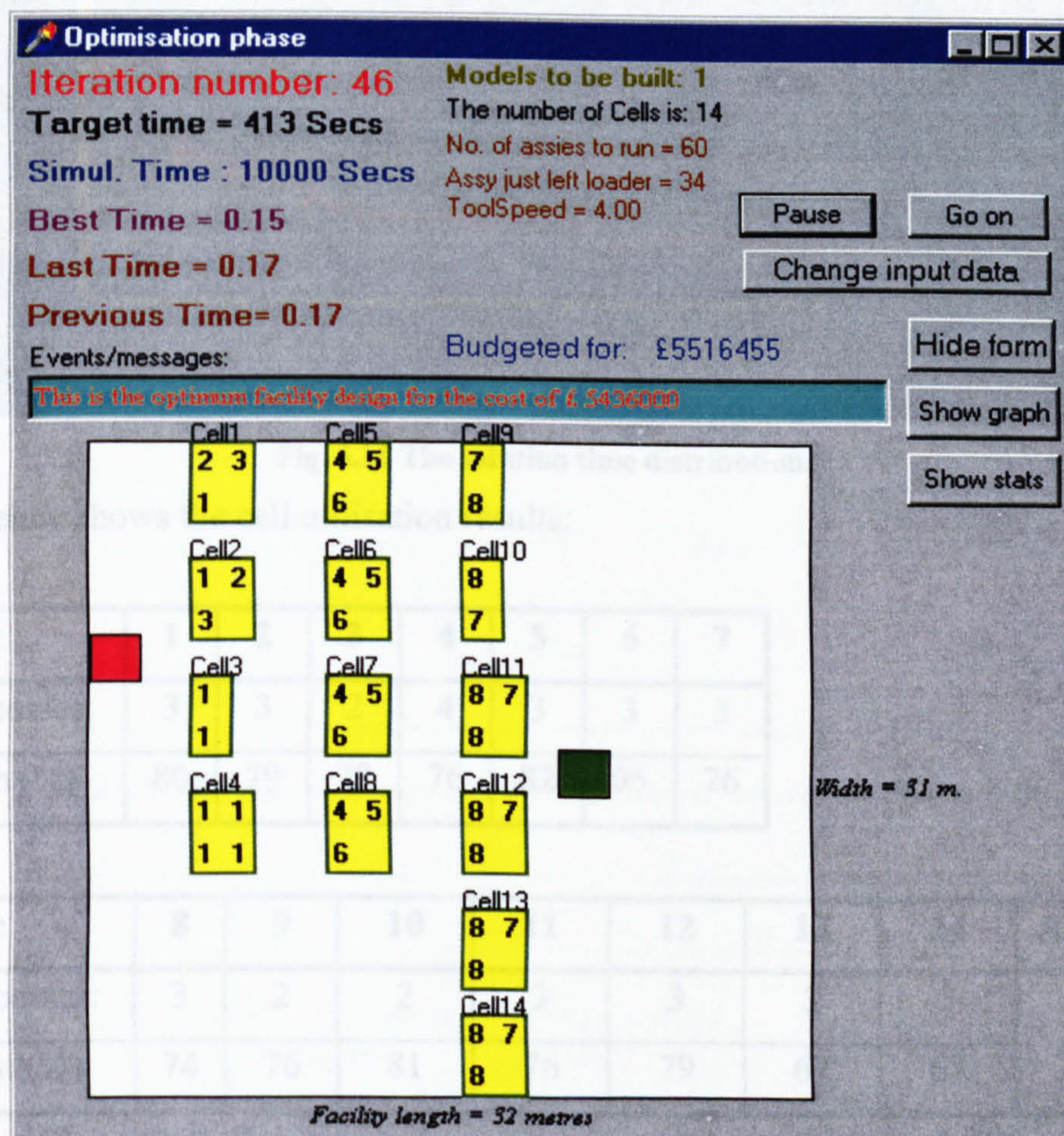


Fig. 5.10 The end result of the third case.

SIMAID's best answer is a feasible solution at the 46th iteration, generating a makespan of 377 seconds for the sample of 20 subassemblies or 6 hours and 54 minutes (40 minutes under target time), generated for a cost of £ 5,436,000 or £ 80,455 beneath budget. The time and pattern to reach the final solution can be seen in Fig. 5.11 below.

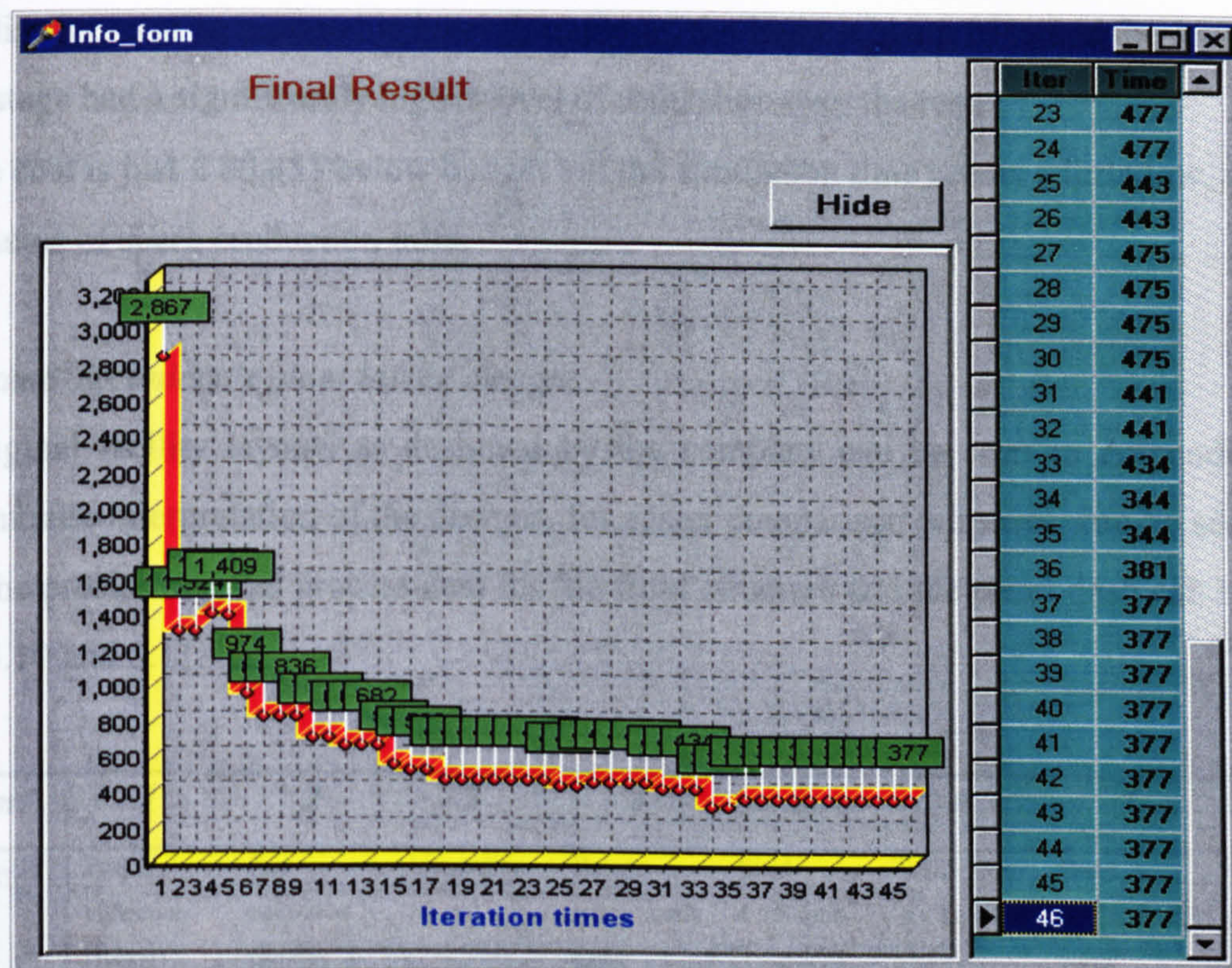


Fig. 5.11 The solution time distribution.

Table 5.9 below shows the cell utilisation results:

Cell:	1	2	3	4	5	6	7
No. of processes:	3	3	2	4	3	3	3
Utilisation (%)	80	79	78	76	82	95	76

Cell:	8	9	10	11	12	13	14	Average util.
No. of processes:	3	2	2	3	3	3	3	77.57
Utilisation (%)	74	76	81	76	79	67	67	

Table 5.9 The results from the simulation.

A first analysis reveals some points of interest:

- There are 8 instances of process 1, two each of processes 2 and 3, four each of processes 4, 5, and 6, six of process 7 and 12 of process 8; The total number of robots and processes is 40;
- In stage I (cells in the first column), even though space in existing cells was available, SIMAID did not place the additional instances of process 1 within cells 1 or 2, preferring to put the 2 extra instances of process 1 into a cell of their own (cell 3);

- Utilisation is at an acceptable level throughout, the highest being 95% and the lowest 67%; no stage had a significantly higher level of utilisation over the rest;
- The cost is just £ 80,455 below budget but the simulation time seems a little low, with 40 minutes of spare production time.

5.3.4 Lamb Technicon's plant layout designs

The original facility layouts as designed by the company can be seen in Appendix C3. A diagrammatic interpretation of the designs, for easier comparison purposes, can be seen in Fig. 5.12. The precise original process data for the front structure and the panel dash can be seen in tables 5.10 and 5.11.

Station:	1	2	3	4	5	6	7	8
Process:	Part collection (fixing)	Part collection (fixing)	Idle station	Rotary index loads parts	Weld 4 x 8 spots	Weld 2 x 11 spots	Weld 2 x 11 spots	Marriage cell (with front dash), Weld 24 spots
Station:	9	10	11	12	13			
Process:	Weld 2 x 11 spots	Weld 2 x 11 spots	Weld 2 x 11 spots	Weld 2 x 11 spots	Unloading cell			

Table 5.10 Original process data for the front structure assembly zone.

Station:	1	2	3	4	5	6	7	8
Process	Manual parts loading	Apply 9 studwelds	Load, apply 2 projection welder bolts, unload	Apply 2 studwelds and 2 Proj. welder bolts	Apply sealant	Apply projection welder	Manual load to fixture	Apply 18 spotwelds
Station:	9	10	11	12	13			
Process	Apply 20 spotwelds	Apply 18 spotwelds	Apply 12 weldspots by static weldgun	Apply 10 weldspots by static weldgun	Move completed dash to buffer			

Table 5.11 Original process data for the panel dash assembly zone.

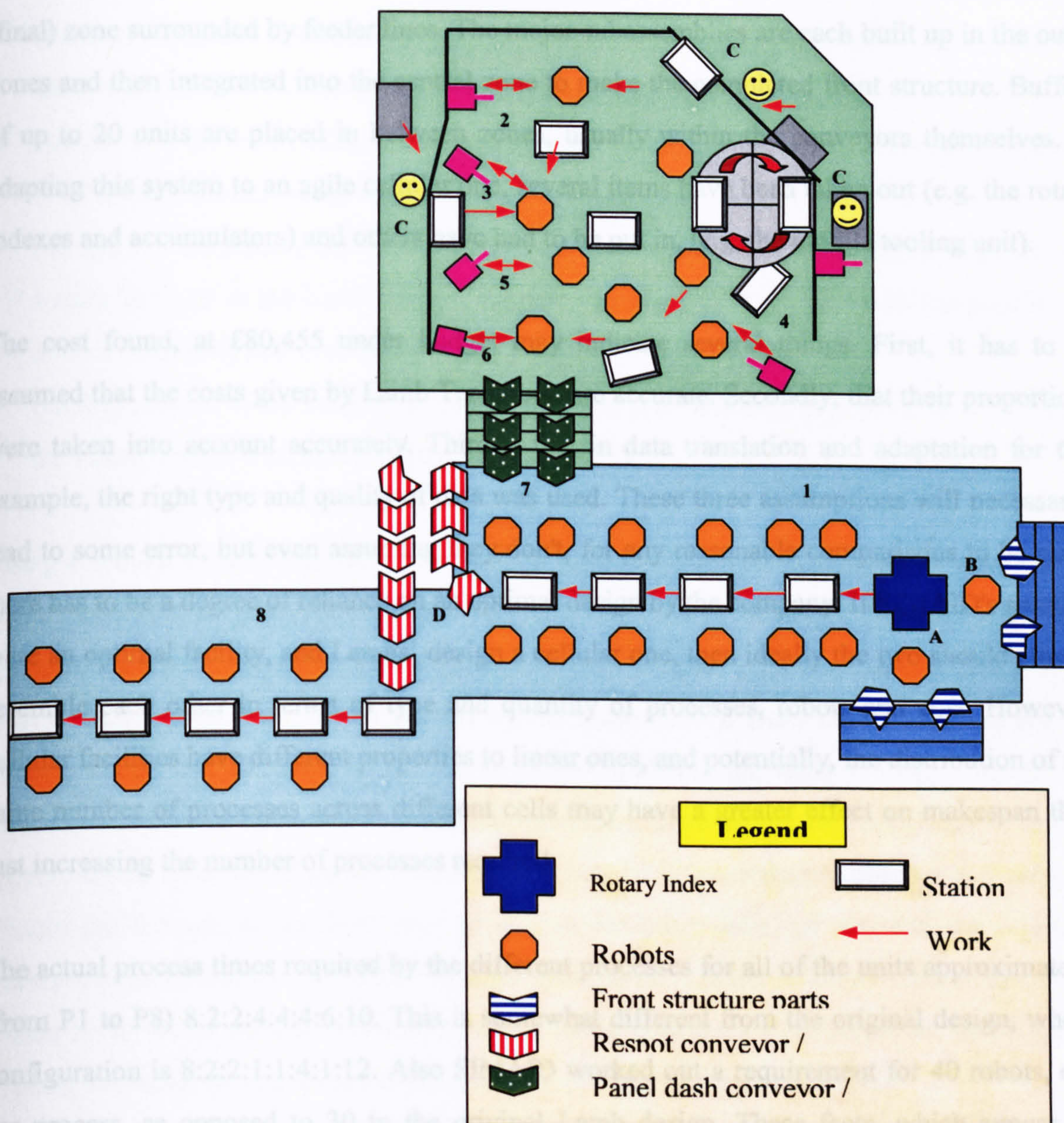


Fig. 5.12 Schematic diagram of the original Lamb design for the plant. The component loading areas are labelled A-C, robotically in the front structure zone and manually in the panel dash zone. The processes are: 1. Spotwelding, 2. Bolting, 3. Studwelding, 4. Sealant application, 5. Projection welding, 6. Spotwelding, 7. The subassembly marriage area, 8. The final combined assembly spotwelding. The area labelled D is the buffer between the marriage station (7) and the respot zone.

5.3.5 Critical analysis and discussion

This example is the most interesting so far because of two unique features:

1. It comprises a cascade-type assembly where one subassembly is joined to another, and
2. There is no real need for any scheduling, as the two subassemblies have to be built and assembled consecutively.

As with earlier designs, this was meant to be a conventional production line, with a central (final) zone surrounded by feeder lines. The major subassemblies are each built up in the outer zones and then integrated into the central zone to make the completed front structure. Buffers of up to 20 units are placed in between zones, usually within the conveyors themselves. In adapting this system to an agile cellular one, several items have been taken out (e.g. the rotary indexes and accumulators) and others have had to be put in, (e.g. the mobile tooling unit).

The cost found, at £80,455 under budget may indicate several things. First, it has to be assumed that the costs given by Lamb Technicon are accurate. Secondly, that their proportions were taken into account accurately. Thirdly, that in data translation and adaptation for this example, the right type and quality of data was used. These three assumptions will necessarily lead to some error, but even assuming they don't, for any reasonable comparisons to be made there has to be a degree of reliance on an optimal design by the company. If SIMAID's solution were an optimal facility, and Lambs' design a cellular one, then ideally the two should closely resemble each other in terms of type and quantity of processes, robots and cost. However, cellular facilities have different properties to linear ones, and potentially, the distribution of the same number of processes across different cells may have a greater effect on makespan than just increasing the number of processes required.

The actual process times required by the different processes for all of the units approximate to (from P1 to P8) 8:2:2:4:4:4:6:10. This is somewhat different from the original design, whose configuration is 8:2:2:1:1:4:1:12. Also SIMAID worked out a requirement for 40 robots, one per process, as opposed to 30 in the original Lamb design. These facts, which appear far removed in likeness to the original design, can actually be explained with a closer look.

- Process 1 is the spotwelding, done by 8 robots of the front structure subassembly in Lamb's layout. The other two are used for loading the parts. SIMAID created 8, which is a perfect match.
- Processes 2 and 3 are the panel dash assembly bolting and studwelding, and the Lamb layout makes provisions for 2 such machines each. SIMAID also created 2 each, which is a perfect match.

- Processes 4 and 5 are the panel dash assembly sealant application and the projection welding. Lamb has again allowed two of them (one each). SIMAID, however, created four each, one in each cell. This appears to be a major discrepancy.
- Process 6 is the panel dash assembly projection welding, which Lamb placed on 4 robots in its design. SIMAID also created 4.
- Process 7 is the mating cell between the panel dash and front subassemblies. There is only 1 such 'process' in the Lamb design (number 7 in Fig. 5.3.2) but SIMAID has placed 6 of them, one in each cell in stage 3.
- Finally, process 8, the combined assembly respot zone welding, is present as 12 robots in the Lamb design, whereas SIMAID only created 10.

Being a cellular system, subassemblies have to go through certain number of cells in order to be processed satisfactorily, and by following a certain order. However, since the facility does not utilise a fixed cycle time, the assemblies are sent to those cells that are first free (i.e. not busy), if any, at the time of the request by the subassembly. Busy cells cannot accept any subassemblies, and there are no buffers anywhere to absorb the waiting ones. If the facility needs to build those units within a definite makespan, there should be enough cells ready to receive them. Even doubling the number of instances of a certain process per cell may not double the throughput, as there may be queues for other cells upfront, and the assembly becomes blocked and the cell locked to other assemblies. A better (if more expensive) method would be to create another cell with a similar set of processes ... and now we can begin to explain the earlier discrepancies in process numbers affecting processes 4-6.

To understand this it is important to realise that if these processes were not present in those cells, the relevant subassemblies could not get completely processed. Due to the nature of the design, agile cellular facilities should not (and SIMAID does not) accept partly processed subassemblies reversing direction to receive further processing. All passage through the cells is strictly and irreversibly unidirectional. Hence, if the requirement is for four process sixes, but the subassemblies also require processes four and five, then the only way to do it (short of creating another stage in between I and II), is to include them in the same cells. A similar discrepancy is the creation of six process seven (the marriage cell). This can be explained by the same principle described above ... Lamb only placed 1 in their design but SIMAID created six, one per cell.

Finally, Lamb placed 12 robots with spotwelders in the respot zone, while SIMAID only created 10. However, on close examination, it can be seen that 4 of the robots (in station 8) are actually being used to transfer the completed panel dash into the front end structure zone, as well as applying 24 spotwelds (6 each). As the other robots are applying 11, or almost twice the number of spots each, this means the transfer cell robots are used for approximately half the time, in effect 'eliminating' the need for 2 of the robots. The adjusted value would actually be 10 working processes, or exactly the quantity SIMAID has created.

The issue of the cost is more perplexing. While within budget, SIMAID created many more robots and processes than in Lamb's version, which should have made facilitisation to the stated budget very difficult. Even allowing for the items not included, the cost should have been far higher. Here are some possible explanations:

- Inaccurate costs, such as those of the site, tooling, cell ancillary or control system;
- Some of the costs were misleadingly applied, such as the number of tooling units;
- The cost table is incomplete, with other necessary entries required in 'real life' but not present here. For example, in the Lamb version a simple conveyor moves the assemblies along, but in an agile cellular facility like the one SIMAID creates would need a more expensive conveyor, together with a central control system for assembly delivery to the cells as well as a cell-level control system for acceptance, processing and dispatching.

Finally, the issue of makespan can be explained by examining what happens in a line. Although industrial simulation will allow designers to balance the stations with an approximate amount of work each, they are all governed by the time taken at the longest station. In Lamb's design for all zones of this example, this is 36 seconds (cycle time). However, there will be many other stations where the time taken is much less - and here is one of the advantages of creating agile facilities. If the processing time in a cell is less than another, the assembly, assuming there is a free cell upstream, will be sent to it before that of its sister cell with a longer total processing time. If applied facility-wise and over a period of time, significant time savings can be made, in this case reducing makespan by as much as 40 minutes per shift.

Lamb's facility has been designed to be productive allowing for 15% downtime, meaning that, allowing for any breakdowns, 15% more production could be potentially achieved. SIMAID managed to meet the makespan requirements in 377 seconds, equivalent to 91% of the 413 seconds target time. Without any modifications to the facility, an approximate 9% more production could be extracted from it. However, when looking at the cell utilisation ratios, the single highest cell utilisation was cell 6, with a value of 95%. This implies that, theoretically (assuming an optimum schedule leading to zero cell blockage/starvation), a further 5% could be extracted from the facility, and when summed to the previous value of 9% the total slack time approximates to Lamb's 15% extra availability figure. In practice this is not usually feasible in facilities that don't resemble balanced transfer lines; the subassemblies need to be processed in whole units and load imbalances mean a higher demand does not translate to a higher utilisation. However, as all of the other cells have a lower utilisation value the probability of a higher production being achieved is good. It is worth bearing in mind, however, that any facility utilisation level approaching the 100% mark, apart from being difficult to implement, is not particularly desirable. A certain degree of slack in the system is beneficial for a number of reasons, primarily as a safeguard against any production volume increase or breakdown allowance.

5.3.6 Conclusion

This example showed that a multi-stage agile facility design could be implemented utilising the SIMAID methodology. After allowing for assumptions, data translation and adaptation, it can be seen that the resulting characteristics bear sufficient similarity to a commercially designed facility to encourage further research in this area.

6. Summary and Conclusion.

6.1 Summary

Due to the increasingly ephemeral nature of manufacturing, layout optimality, even where known, is of relative importance and not necessarily desirable at any cost. This is because optimality is determined for specific model mixes and volumes, and is lost if any change is required. The cost, in terms of money and time, of continually attempting to attain an optimal layout far outweigh the benefits derived from it. Hence, it is more practical (and economical) to be able to quickly devise a 'good' layout and operate it until changing conditions require a layout revision. The 'goodness' of the generated solution (derived layout) can be judged by its proximity to the stated aims and objectives.

A novel methodology has been presented here which generates facility layouts combining elements from group technology, genetic algorithms and iterative improvement simulation similar to Tabu search. The methodology is in three stages, with the first stage leading to an initial (infeasible) cell layout to aid the search direction process of the next two stages. This is done by utilising model-specific production and volume data to generate a layout based on individual process volume and sequence.

The second stage schedules the units to be assembled in the facility to ensure the minimum use of the improvement process, as well as the highest cell utilisation and the best space use. The scheduling is carried out by utilising a $2n$ subassembly number genetic algorithm, n being the number of cells within the current layout, to ensure a rapid result.

The third stage is the improvement process, which consists of two separate phases, the first used to reach a feasible solution and the second to improve upon it. Each phase utilises a different goal for achieving its aims, the first using cell utilisation and process request frequency, the second utilisation with space, cost and makespan. From this point onwards, the second and third stages are operated cyclically until no more improvements can be achieved.

The methodology has been tested on three industrial case studies. Given the limitations imposed by the different data formats and facility aims, the methodology has given good answers. All three designs were generated with a similar level of processes, use of space and to

budget to the supplier's own designs, taking into consideration the different formats. Case study 1 had 23 robots and processes to the original design of 21. Case study 2 had twice as many (16 as opposed to eight) as the original, but the two designs are not directly comparable due to the type of processes used; on further examination, taking cell utilisations and lack of buffers into account, the two designs differ little. For case study three SIMAID generated a requirement for 40 robots to the original's 30; however, in the original design, some of those were not process-related but transfer robots, the rest of the excess being due to the presence of multi-process machines.

6.2 Conclusions

The methodology has been shown to develop good solutions for the complex problem of cellular design for future multi-process, mixed-model agile facilities. The novel combination of process-led part characterisation of the input data, the guided initial layout creation, the genetic-algorithm-based subassembly scheduling and the subsequent solution improvement iteration cycles combine into a robust set of algorithms for the effective development of future cellular facilities. This work has demonstrated that the methodology incorporated as the program SIMAID is able to devise facility layouts, comparable to conventional plant layouts from existing production data, to agile principles. Nevertheless, the results proceeding from the case studies could not be considered to constitute examples of *agile layouts*, due to the fact that no such layouts have been built to date to be compared with. Hypothetical cases of such layouts have been devised and their data used, but these have limited usefulness and have not been incorporated here. However, considering that the aim is to rapidly generate good cellular layouts from production and volume data for subsequent customisation in commercial simulators, the main conclusions are:

- The main advantage of this guided search method is its ability to find initial (unfeasible) solutions in the neighbourhood of the global optimum cone, thus cutting down on the number of improvement iterations required to approach it;
- The second stage of the methodology involves using genetic algorithms to schedule the subassemblies by separating them into manageable individual schedules, and schedule them sequentially. The method (described in fig 3.18) is designed to circumvent the problems associated with long schedules. Although not optimal, this approach takes

advantage of the cellular nature of the layout to avoid being overwhelmed by the exponential nature of the problem;

- The dual phase (infeasible → feasible, feasible → optimal) solution mechanism of the third stage (fig 3.20) works well at finding good solutions in relatively few iterations, even if the optimum is never reached; SIMAID found good solutions (similar in context) to the industrially-derived problems in under 20 iterations for two case studies, and under 50 iterations for the third one.
- The graphical user interface (GUI) format aids the understanding of the underlying issues involved and allows quick and easy access to data for modification purposes.

As the supplier's designs have been optimised and validated by discrete event simulation, it indicates that the methodology has a valid base for further development.

6.3 Further work

Several improvements, both to the methodology and to the ease of use of the program could be investigated, and are listed below. These have been classified into 'method' and 'features', to distinguish between them.

6.3.1 Method:

- Process placing within cells – further study into the factors affecting the placing of dissimilar processes into cells due to sequence and stage requirements;
- Scheduling – further investigation into the decisions relating to the best situation for scheduling to occur between the second and third stage iterations; at present this occurs for every major change within the facility;
- Scheduling – potential use of polygamy as a reproductive mechanism, utilising only the best chromosomes;
- Process optimisation – the incorporation of a 'third' loop, allowing the program to find higher quality solutions through the rationalisation of processes and space within cells of different stages;

6.3.2 Features:

- Use of space – a more exact evaluation as to the requirements of space-devouring MHS 'corridors' from the first to later stages in the layout;

References

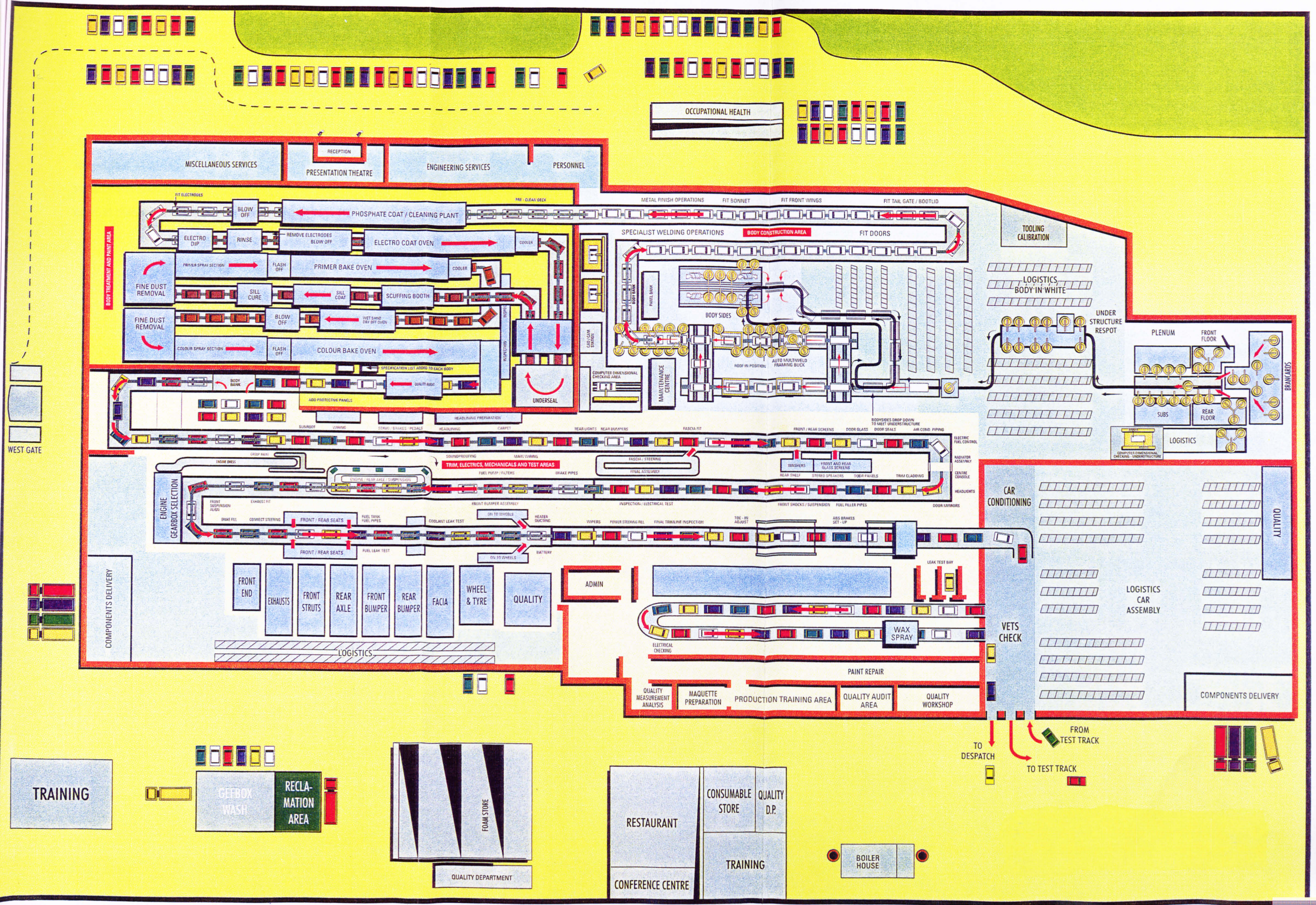
1. Anon, 1989. Ringing the changes at Emden. *The Industrial robot*, 16 (4), 212-215, 1989. IFS Publications, 0143-991X.
2. Askin, R.G., and Chiu, K.S., 1990. A graph partitioning procedure for machine assignment and cell formation in group technology. *International Journal of Production Research*, 28, No. 8, 1555-1572.
3. Beaulieu, A., Ghabi, A. and Ait-Kadi, 1997. An algorithm for the cell formation and the machine selection problems in the design of a cellular manufacturing system. *International Journal of Production Research*, 35, No. 7, 1857 - 1874.
4. Chen, C-W, and Sha, D.Y., 1999. A design approach to the multi-objective facility layout problem. *International Journal of Production Research*, 37, No. 5, 1175 - 1196.
5. Chiang, WC., and Kouvelis, P., 1996. An improved tabu search heuristic for solving facility layout design problems. *International Journal of Production Research*, 34, No. 9, 2565-2585.
6. Chittratanawat, S., and Noble, J.S., 1999. An integrated approach for facility layout, P/D location and material handling system design. *International Journal of Production Research*, 37, No.39, 683-706.
7. Conway, D.G. and Venkataramanan, M.A., 1994. Genetic search and the dynamic facility layout problem. *Computers Operations Research*, 21, No. 8, 955-960.
8. Davis, L.. Job shop scheduling with genetic algorithms. *Proceedings of the International Conference on Genetic Algorithms and Applications*, 1985, pp. 136-140.
9. Falkenauer, E., et al. A GA for job shop. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991, pp 824-829.
10. Farish, M., 1995. Fast track. *Engineering*, October 1995.
11. Gau, K-Y, and Meller, R.D., 1999. *International Journal of Production Research*, 37, No.16, 3739-3758.
12. Gen, M., et al. Solving job shop scheduling problems by genetic algorithms. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 1994, pp 1577-1582.
13. Geoff Barton, *Automotive News Europe*, 1 February 1999, P.12.
14. Goldberg, D. Genetic algorithms in Search, Optimisation and Machine Learning, 1989. Addison-Wesley, Reading, Massachusetts.
15. Ham, I., Hitomi, K., and Yoshida, T., 1985. Group technology – Applications to production management. Kluwer-Nijhoff Publishing, USA. Lee, S-D., and Chen, Y-L., 1997. A weighted approach for cellular manufacturing design: minimising intercell movement and balancing workload among duplicated machines. *International Journal of Production Research*, Vol. 35, No. 4, 1125-1146.
16. Hao, g., Shang, J.S. and Vargas, L.G., 1995. A neural network model for on-line control of flexible manufacturing systems. *International Journal of Production Research*, Vol. 33, No. 10, 2835-2854.
17. Holweg, M., and Jones, D.T., 2001. The build-to-order challenge. *Automotive World*, January/February edition, 40-45.
18. Huntley, C.L., and Brown, D.E., 1991. A parallel heuristic for quadratic assignment problems. *Computers & Operations Research*, 18, 275-289.
19. Irani, S.A., Cohen, P.H. and Cavalier, T.M., 1991. PED, Vol. 53, Design, analysis and control of manufacturing cells, ASME 1991.
20. Jayakrishnan, N. and Narendran, T.T., 1998. CASE: A clustering algorithm for cell formation with sequence data. *International Journal of Production Research*, Vol. 36, No. 1, 157-179.

21. Jayakrishnan, N. and Narendran, T.T., 1999. ACCORD: a bicriterion algorithm for cell formation using ordinal and ratio-level data. *Int. J. Production Research*, Vol.37, No.3, P539-556, 1999.
22. Johnson, J. and Picton, K., 1999. Concepts in artificial intelligence. Butterworth-Heinemann, Open University publications, Milton Keynes. ISBN 0-7506-2403-5.
23. Khoo, LP and Ong, NS, 1998. PCB assembly planning using genetic algorithms. *International Journal of Advanced Manufacturing Technology*, No.14, 363-368.
24. Kidd, P., 1996. Agile manufacturing: A strategy for the 21st century. IEE, manufacturing division, proceedings from the colloquium held on 28 March 1996 at Cranfield University. UK ISSN 0963-3308.
25. Kusiak, A. and Heragu, SS, 1987. The facility layout problem. *European Journal of Production Research*, No. 29, 229 - 251.
26. Kusiak, A., and Cheng, C., 1991. Group technology: Analysis of selected models and algorithms. *PED, Vol. 53, Design, analysis and control of manufacturing cells. ASME 1991*.
27. Kusiak, A. and He, D.W., 1997. Design for agile assembly: an operational perspective. *International Journal of Production Research*, 35, No. 1, 157 - 178.
28. Lawler, E., 1976. Combinatorial optimisation: Networks and matroids. Holt, Rinehart and Winston Editors, USA.
29. Lee, S.D., and Chen, Y.L., 1997. A weighted approach for cellular manufacturing design: minimising intercell movement and balancing workload among duplicated machines. *International Journal of Production Research*, 35, Vol. 35, No. 4, 1125-1146.
30. Lozano, S., Guerrero, F. and Onieva, L., 1999. Cell design and loading in the presence of alternative routing. *International Journal of Production Research*, Vol. 37, No. 14, 3289-3304.
31. Mansouri, S.A., Moattar H., and Newman S.T., 2000. *International Journal of Production Research*, Vol. 38, No. 5, 1201-1218.
32. Masuyama, A, 1995. Idea and practice of flexible manufacturing systems of Toyota. Flexible Manufacturing Systems: Recent developments. Elsevier Science B.V.
33. Mattucci, M., Rossi, C., and Cuddy, R., 1994. International body engineering conference: Papers. IBEC '94, Detroit, Michigan, Vol. 8.
34. McMullen, P.R. and Frazier, G.V., 1988. Using simulated annealing to solve a multi-objective assembly line balancing problem with parallel workstations. *International Journal of Production Research*, 36, No. 10, 2717-2741.
35. Nair, G.J., and Narendran, T.T., 1998. CASE: A clustering algorithm for cell formation with sequence data. *International Journal of Production Research*, 36, No. 1, 157-179.
36. Nicholson, T., 1971. Optimisation in Industry: Volume 2, Applications. Chicago: Aldine-Atherton publishers.
37. Proth, J.M., and Vernadat, F., 1991. COALA: A manufacturing layout approach. *PED, Vol. 53, Design, analysis and control of manufacturing cells, ASME 1991*.
38. Rajamani, D., Singh, N., and Aneja, Y.P., 1990. Integrated design of cellular manufacturing systems in the presence of alternative process plans. *International Journal of Production Research*, 28, No. 8, 1541 - 1554.
39. Rajasekharan, M., Peters, B.A. and Yang, T., 1988. *International Journal of Production Research*, 36, No. 1, 95-110.
40. Rao, H.A., Pham, S.N., and Gu, P, 1999. *International Journal of Production Research*, 37, No. 3, 557-580.
41. Retrospective, *Automotive News Europe*, 21 December 1998, P21.
42. Salum, L., 2000. The cellular manufacturing layout problem. *International Journal of Production Research*, 38, No. 5, 1053-1069.

43. Seifoddini, H. and Djassemi, M., 1995. *Journal of Manufacturing Systems*, 14, 35-44.
44. Richard Johnson, *Automotive News Europe*, 15 February 1999, P.10.
45. Sawik, T.J., 1995. Scheduling flexible flow lines with no in-process buffers. *International Journal of Production Research*, 33, No. 5, 1357-1367.
46. SU, C-T, and Hsu, C-M., 1998. Multi-objective machine-part cell formation through parallel simulated annealing. *International Journal of Production Research*, 36, No. 4, 2185-2207.
47. Tanaka, H, and Yoshimoto, K., 1993. Genetic algorithms applied to the facility layout problem. Technical report, Department of Industrial Engineering and Management, Waseda University, Tokyo.
48. Vakharia, A.J. and Chang, Y-L., 1997. Cell formation in group technology: a combinatorial search approach. *International Journal of Production Research*, 35, No. 7, 2025-2043.
49. Wan, Y.W., 1995. Which is better, off-line or real-time scheduling? *International Journal of Production Research*, 33, No. 7, 2053-2059.
50. Wang, W., and Brunn, P., 2000. An effective genetic algorithm for job shop scheduling. *Proceedings of the ImechE*, Vol. 214, No. B4.
51. Welgama, P.S. and Gibson, P.R., 1996. An integrated methodology for automating the determination of layout and material handling system. *International Journal of Production Research*, 34, No. 8, 2247 - 2264.
52. Wild, R., 1995. *Production and operations management*. Cassel Educational Ltd., ISBN 0-304-44077-9.
53. Wu, N., 1998. A concurrent approach to cell formation and assignment of identical machines in group technology. *International Journal of Production Research*, 36, No. 8, 2099 - 2114.
54. Zhang, C., et al. A genetic algorithm of evolving job-shop scheduling problem. *Chin Journal of Electronics*, 1995, 4(1), 48-52.

Appendix A1

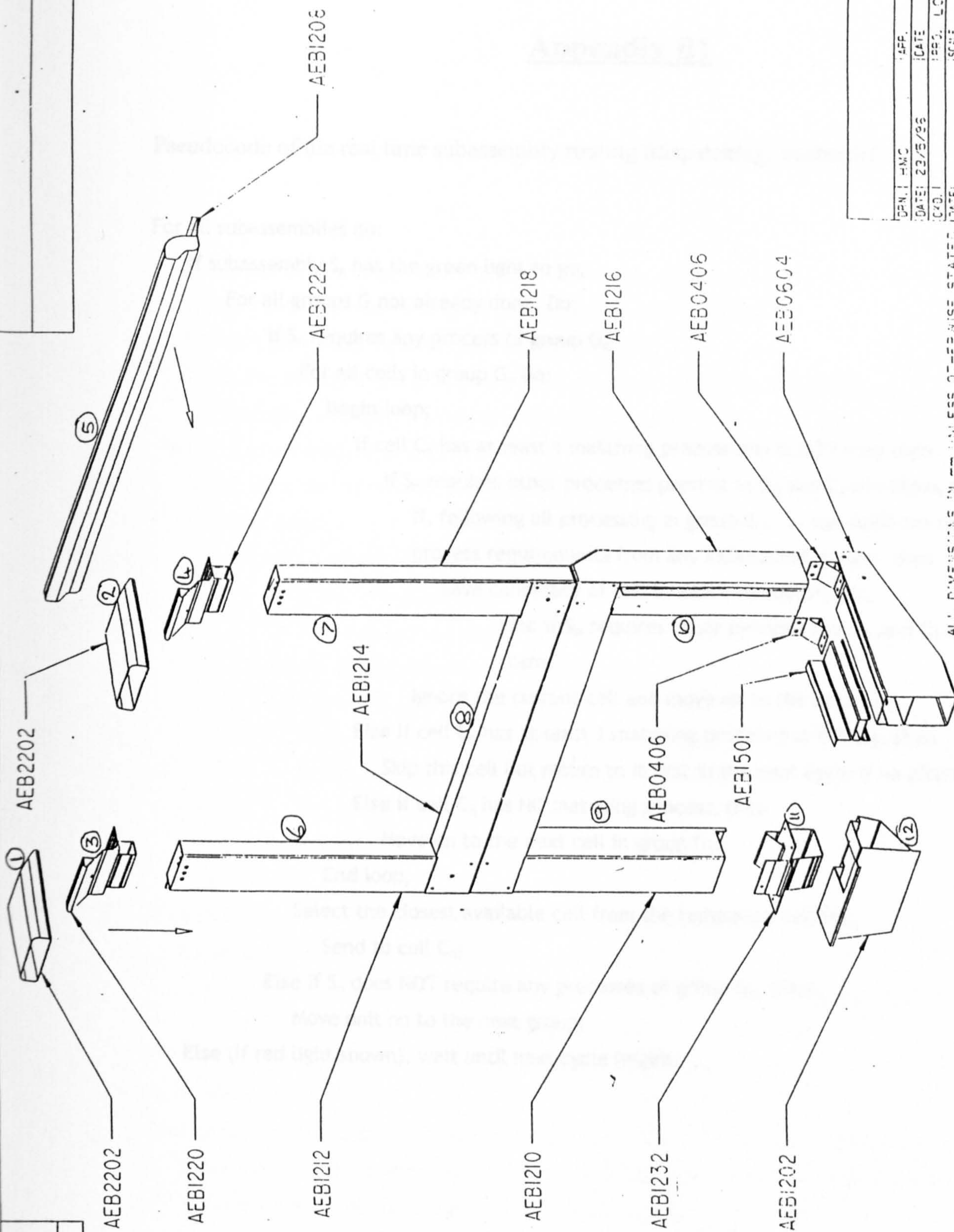
Schematic layout of a current factory, showing the conventional line assembly system.



Appendix A2

A typical spaceframe-based subassembly designed for the SALVO project, showing the individual aluminium extrusions, which comprise part of the left bodyside of the prototype vehicle (overleaf). All of the individual components are then fitted onto a mobile tooling and carried through the cellular production facility for processing.

IV



WATER DIVERSITY IN THE WILSON CREEK WATERSHED

[illegible]

A3M

DEF. NO.

SHEET NO. 1335

Appendix B1

Pseudocode of the real time subassembly routing (dispatching) controller:

For all subassemblies do:

 If subassembly S_n has the green light to go;

 For all groups G not already done, Do:

 If S_n requires any process of group G_n ;

 For all cells in group G_n Do:

 Begin loop;

 If cell C_n has at least 1 matching process and is NOT busy then

 If S_n requires other processes present in G_n and C_n has them, then

 If, following all processing in group G_n , S_n can fulfil the rest of its process requirements from any subsequent groups, then

 Save candidacy of cell S_n into a temporary list;

 Else if S_n requires other processes of G_n and C_n does NOT have them,

 Ignore the current cell and move on to the next one;

 Else if cell C_n has at least 1 matching process but IS busy, then

 Skip this cell but return to it first in the next cycle if no alternatives;

 Else if cell C_n has NO matching process, then

 Move on to the next cell in group G_n ;

 End loop;

 Select the closest available cell from the temporary cell list;

 Send to cell C_n ;

 Else if S_n does NOT require any processes of group G_n , then

 Move unit on to the next group;

 Else (if red light shown), wait until next cycle begins;

Appendix B2

Pseudocode of the first cycle of the improvement iterations in stage 3.

If p is not in the same stage as the facility's most queued after cell, C_q , then

Find the cell in the same stage as C_q with the lowest utilisation;

Subtract the processes of this cell from those of C_q ;

Select the single busiest process of these and place it into the cell lacking it;

Exit iterative procedure; (*Don't do any of the rest*)

Else (*If it is in the same stage*)

For all the cells in group G_p , Do:

Begin loop;

If cell C_n has process p present, then

If C_n has $< RC_{\max}$ processes, then

Save cell S_n in a temporary list;

Else (*i.e. cell C_n does not have p present*)

If C_n has $< RC_{\max}$ processes, then

Place p in the first available space within this cell;

Exit procedure; *{the process has been placed, thus the next simulation iteration can begin}*

Else go on to next cell;

End loop;

If no selection has been made and a temporary cell list exists, then

For all cells in the temporary list of p -containing cell, do:

Find and select the cell that has had the least use during the simulation run;

Add the process p to the first available space within this cell;

Exit procedure; *{the process has been placed, thus the next simulation iteration can begin}*

Else (*If no selection has been made as no cell has $< RC_{\max}$ processes, i.e. all cells are full*)

If there is sufficient space below the last cell in stage (within the predefined area), then

Create a new cell C_{n+1} beneath the last cell;

Place the process p in the first slot;

Exit procedure; *{the process has been placed, thus the next simulation iteration can begin}*

Else (*there is insufficient space due to facility size limitations*)

Create a new cell C_{n+1} in a new column, immediately after the last one, still within the group G_p 's zone, but before the next group (or the unloader/framing station);

Place the process p in the first slot;

End of procedure;

Appendix B3

Pseudocode of the second cycle of the improvement iterations in stage 3.

If any cell in the facility has zero utilisation levels, then:

Remove this cell;

Shift the remaining cells up/left within the stage and/or other stages closer to the loader
if affected, exit this procedure, and run the next simulation iteration;

Else (*If any cell has a low utilisation, but not zero, defined as the minimum acceptable value by the user as an input or < 50% of the facility average by default*)

Find the least busy cell, C_{lb} , and the next least busy cell, C_{nlb} , in the same stage;

If there are any processes p in C_{lb} which are not present in C_{nlb} , then

place p in C_{nlb} and destroy C_{lb} ;

Exit procedure;

Else (*If there are no such processes*)

If there are sufficient spaces in the rest of the cells of the stage to evenly distribute all
of the processes present in C_{lb} ,

Do so,

Destroy the empty cell, exit the procedure, and run the next simulation iteration;

Else (*If not*)

If the shortest and second shortest process value of this stage combined are less than the
single longest process value of any stage, then

Find the cell in this stage with the two lowest process times,

Remove a robot and add a toolchanger for these 2 processes;

Exit procedure;

Else

If the tooling speed is set above minimum,

Lower the tooling speed by 10%;

Else exit procedure;

Do the next simulation run;

If feasible,

return to the top for the next improvement iteration;

Else record the last solution in a Tabu list;

If the unfeasible solution is dramatically worse than the previous one,

leave the feasible solution improvement cycle and return to the previous one;

Else undo the change and return to the top of this cycle;

End of procedure;

Appendix B4

Pseudocode of the third and final cycle of the improvement iterations in stage 3.

For each stage do:

 If there is a single- or double-processed, low utilisation cell C_{sd} in stage $S1$,

 If there are any cells in the *next* stage with $< RC_{max}$ processes,

 For all such cells do,

 Begin loop:

 If the number of processes present in these cells + those of $C_{sd} \leq RC_{max}$,

 And if the total processing time of this cell + that of C_{sd} is less than that of the highest in the facility,

 If there is more than one such cell,

 Select the least busy cell and place the process (es) of C_{sd} in it;

 Destroy the original cell (C_{sd}) and shift any neighbouring cell, if required;

 Run a 'feasibility' or trial simulation iteration to ensure all processing requirements can be fulfilled;

 If all processing requirements CAN be fulfilled,

 Exit the procedure and run the next simulation iteration;

 Else, return to original layout and exit this loop;

 Else, do the same above loop with the only such cell;

 Else exit this loop;

 End loop;

 Else if there are any cells in the *previous* stage with $< RC_{max}$ processes,

 Repeat above loop with this stage;

Else exit procedure (*nothing can be done anymore*);

Appendix C1

Details of the input data and Lamb Technicon’s optimised layout for case study 1.

Lamb Technicon UKA **UNOVA** Company**Facsimile**

Please reply to:

☐ Lamb Technicon UK
Hampstead Avenue
Mildenhall,
Suffolk. IP28 7RE
England

☒ Lamb Technicon UK
22/23 Monkspath Business Park
Shirley, Solihull
West Midlands, B90 4NZ
England

Department: ENGINEERING

Direct Dial: 0121-733-4265

Local Fax: 0121-733-1070

Attn. of: DR. K. W. YoungFrom: J. KOWALEWSKYCompany: WARWICK UNIVERSITYDate: 18 August 1998Telefax: 01203 524 307No. Of Pages: 3 including headerSubject: SIMAD

Our Ref.: _____

Copies: _____

PLEASE RUN THE FOLLOWING DATA ON YOUR
SIMAD PROGRAM FOR TRIAL PURPOSES. THE RESULTS
MAY BE COMPARED TO THE CONCEPT WE HAVE ALREADY
DEVELOPED.

PLEASE CALL ME SHOULD YOU REQUIRE FURTHER
INFORMATION OR FEEL THAT A VISIT WOULD BE OF
BENEFIT.

BEST REGARDS

John Kowalewski

LAMB

To: Dr. K.W. Young
 From: J. KOWALEWSKY
 Date: 17 AUGUST 1998
 Subject: SIMAID

1/2



FLOOR SPACE: 14 x 38 METRES. (FOR EQUIPMENT & PARTS STORAGE)

SHIFTS : 3

HOURS/DAY : 19.8

DAYS/WK : 5

DAYS/YEAR : 233

NET/VOL YEAR : 180,000

MINIMUM EFF: 90% - Overage

PROCESS (ASSEMBLY STAGES)

STAGE 1 = 10 PARTS - LEAD TIME 33 SEC'S (MANUAL)

STAGE 2 = 5 PARTS - LEAD TIME 21 SEC'S (MANUAL)

STAGE 3 = 3 PARTS - LEAD TIME 20 SEC'S (MANUAL)

STAGE 4 = 2 PARTS - LEAD TIME 18 SEC'S (AUTOMATIC)

STAGE 5 = 2 PARTS - LEAD TIME 18 SEC'S (AUTOMATIC)

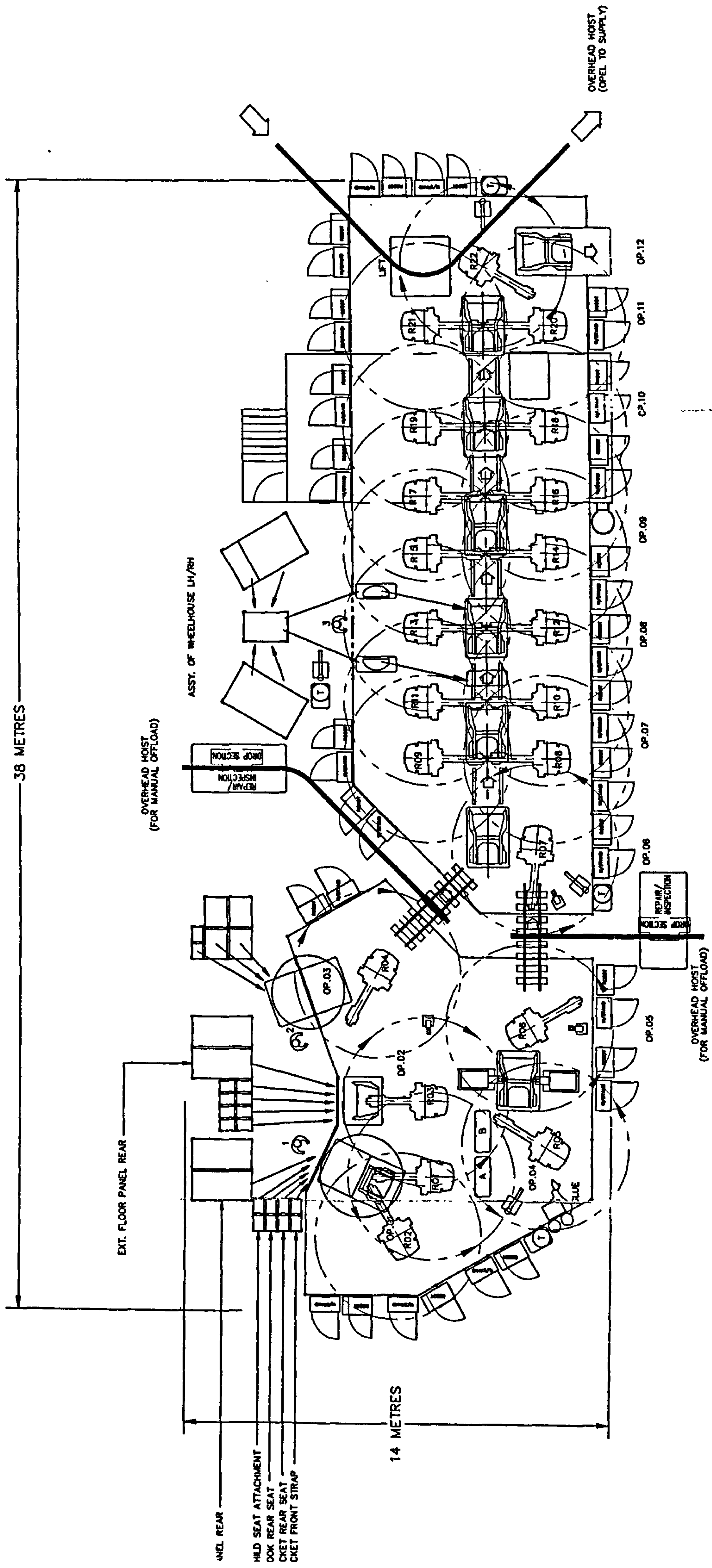
STAGE 6 = 3 PARTS - LEAD TIME 22 SEC'S (AUTOMATIC)

WELD SPOTS PER ASSY: 364 - MINIMUM 12 GUN DESIGNS

WELD BELTS PER ASSY: 16 - MINIMUM 3 TYPES

WELD THRU' SEALANT : 15 cc (1 application ϕ 4mm x 1200 per

Stage 4.



Appendix C2

Details of the input data and Lamb Technicon’s optimised layout for case study 2.



Date: 05 FEBRUARY 1999

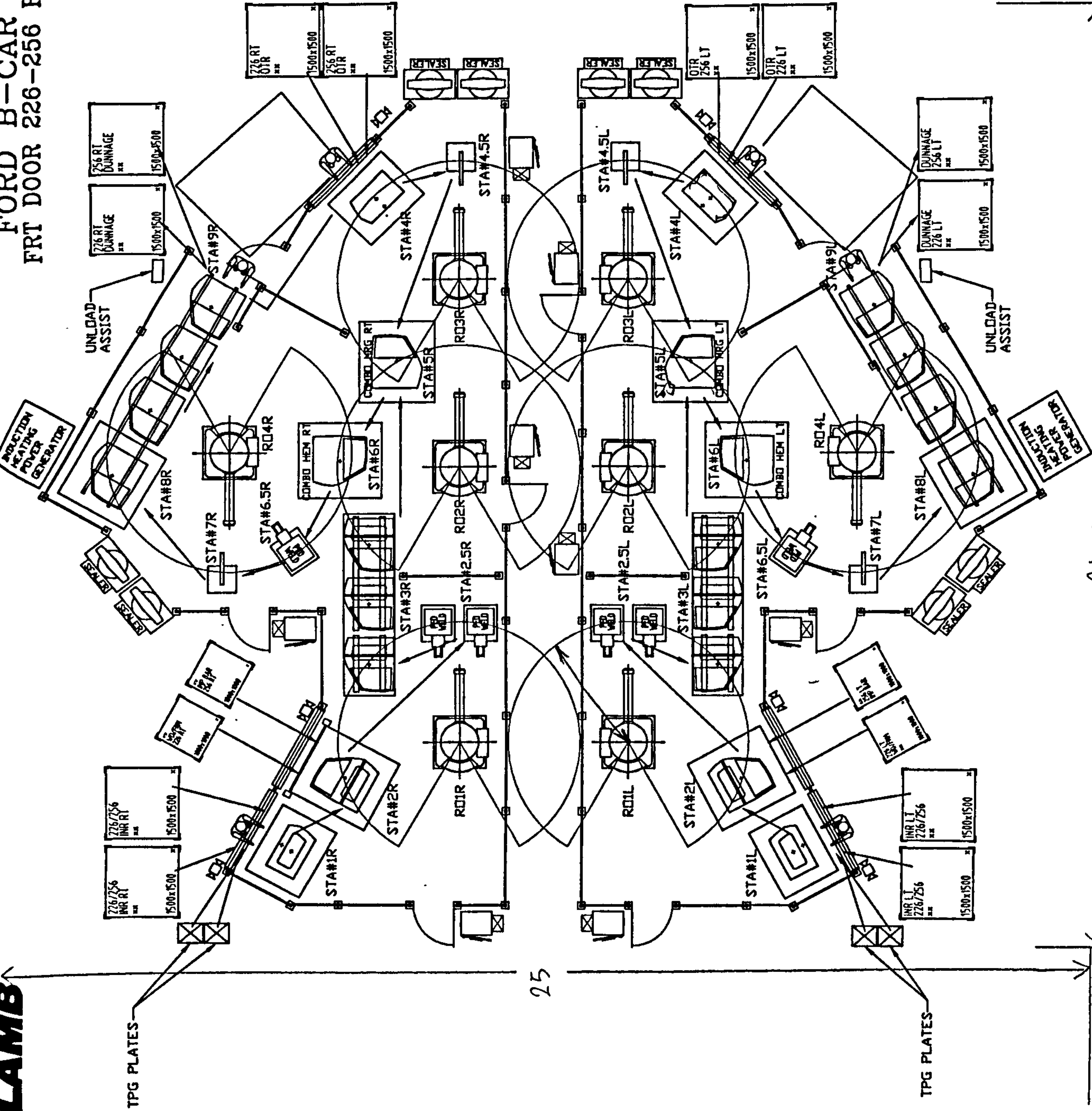
Currency: EURO

Rate: 1.4450

[NOT REQUIRED TO FILL IN

REVISION COLUMN CODES

10-6-24-1



25

PROCESS FLOW

STN No 1L/R	STN No 2L/R	STN No 2.5	STN No 3L/R	STN No 4L/R	STN No 4.5L/R	STN No 5L/R	STN No 6L/R	STN No 6.5L/R	STN No 7L/R	STN No 8L/R	STN No 9L/R
LOAD 3 PARTS MULTIWELD 4 SPOTS	LOAD 3 PARTS MULTIWELD 10 SPOTS	ROBOT REPOST 6 SPOTS	BUFFER	LOAD SKIN	ROBOT APPLY HEM ADHESIVE	MARRIAGE AND PRE-HEN	SERVO HEMWER	ROBOT RESPOT 13 SPOTS	ROBOT APPLY COSMETIC SEALER	ROBOT INDUCTION CURING	OFFLOAD

6712 926

LEGEND

- HANDLING
- WELDING
- HANDLING/FED. WELD
- ROBOT WELDING/
HANDLING
- MIG WELDING
- STUD WELD
- ADHESIVE/SEALANT
- TUCKER STUDWELDING
- PROJECTION WELDING
- PEDESTAL WELDER
- DATE STAMP
- ROBOT GEOMETRY
WELD FEATURE
- HOLDING FEATURE
- DEDICATED
GEOMETRY
WELD FEATURE
- ROBOT/DEDICATED
GEOMETRY
WELD FEATURE
- MANUAL
WELD FEATURE
- RESPOT FEATURE
- STILLAGE
- PIPE RAILS
- OPERATOR (+No.)
- MACH. ASSIST

EQUIPMENT REQUIREMENT

OUTPUT MAIN ASSEMBLY LINE	94 PARTS PER HOUR
EFFICIENCY/AVAILABILITY	
CYCLE TIME	30 SECONDS
NO OF SHIFTS	
FLOOR AREA	
NO OF OPERATORS	4
NO OF SPOTWELDS	35
NO OF PROJECTION WELDS	
NO OF STUDS	
NO OF STUD GUNS	
NO OF GRIPPERS	
NO OF PORTABLE WELD GUNS	
NO OF ROBOTS	8
DATE STAMPING OPERATION	

Lamb Technicon UK

FACILITIES LAYOUT

NOTICE

THIS DRAWING IS THE PROPERTY OF
LAMB TECHNICON UK
A LAMB COMPANY
IT IS NOT TO BE REPRODUCED
OR USED IN ANY MANNER WITHOUT
THE WRITTEN PERMISSION OF LAMB
TECHNICON UK

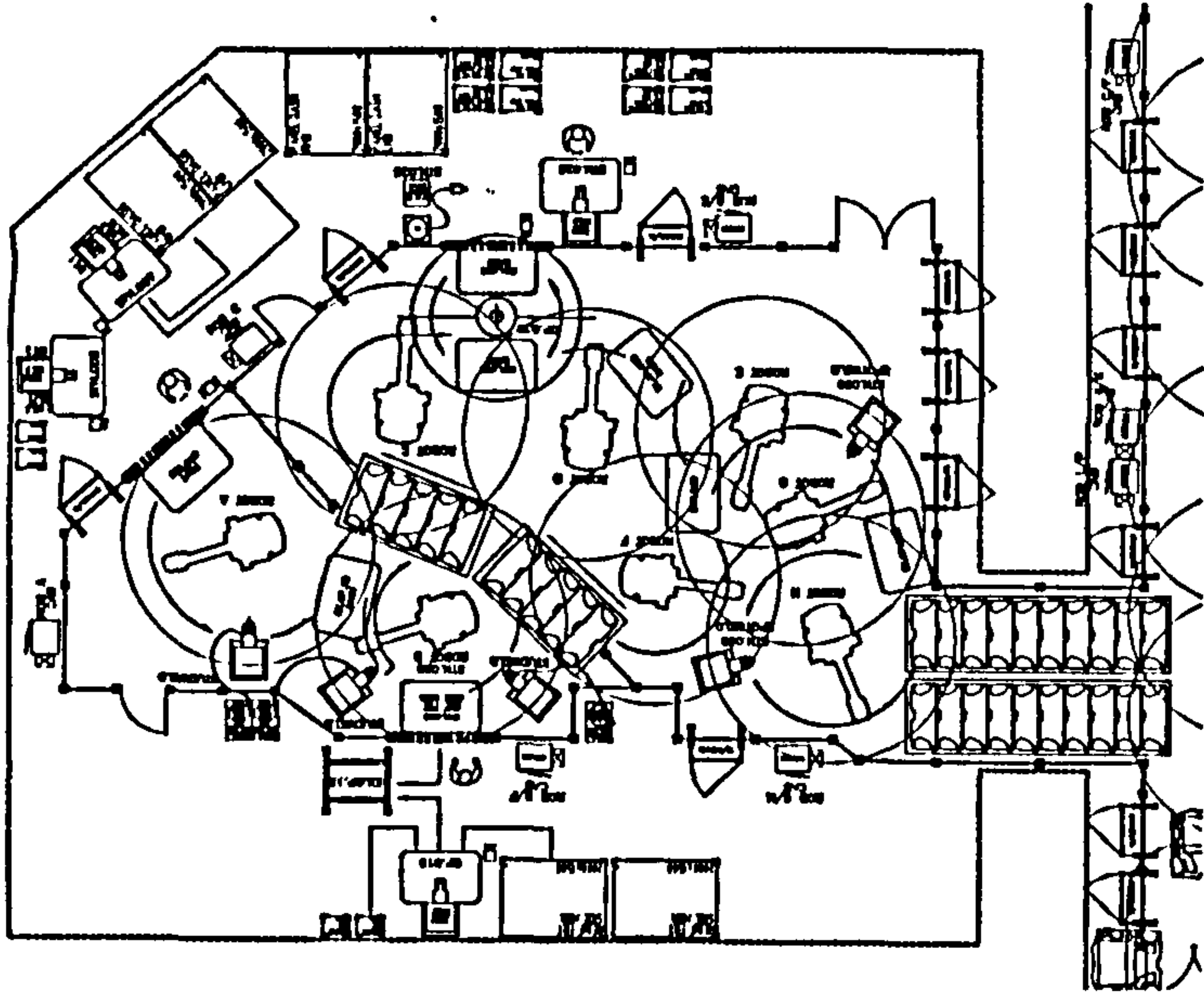
LAMB CONCEPT B-CAR PROGRAM FRT DOOR 226-256 RH/LH COLOGNE	FORD	PW-1679
CD	10	10
KN-0A-FD	AO	

Appendix C3

Details of Lamb Technicon's optimised layout for case study 3, showing both the front and panel dash structures as well as the overall plant.

B-CAR
PANEL DASH ASSEMBLY IE No. 9B

IE No. 9B ZONE 4
DASH
ASSEMBLY



PROCESS FLOW

STN.005	STN.010	STN.015	STN.020	STN.025	STN.026	STN.030	OP.030
MANUAL LOAD TO FUTURE - OPERATOR LOADS 1 PANEL & 3 BELTS TO W/C 1. ACTIVATE W/C OPERATOR UNLOADS ASST TO W/C 2 PLUS 3 BELTS MANUALLY UNLOADS ASST TO OP-10	ROB A COLLECTS EXOP 08 & POSITIONS TO STATION 1. APPLY (1) STUDS. ROB A PLACES ASST ON INTER- OF RACK OP-10	MANUAL LOAD TO PROJ W/C - OPERATOR LOADS 1 PANEL PLUS 2 BELTS ACTIVATE W/C PLACE ON INTER-OF RACK OPERATOR TO UNLOAD ASST TO OP-10	ROB B COLLECTS ASST FROM EXOP 10 & POSITION TO STATION APPLY (2) STUDS PLACE ASST ON INTER-OF RACK POSITIONS TO STATION 2 APPLY (2) BELTS ROB B PLACES ASST ON INTER-OF RACK	MANUAL LOAD TO BLOCK - OPERATOR LOADS 1 PANEL APPLY SEALANT.	OPERATOR LOADS 2 PANELS ACTIVATE PROJ WELDER.	MANUAL LOAD TO FUTURE - OPERATOR LOADS 3 PANELS	ROB C COLLECTS DASH EXOP 30 LOCATE TO FUTURE - ROB C WELD (8) SPOTS ROB D WELD (10) SPOTS ROB E PLACE ASST ON INTER-OF RACK

PROCESS FLOW

STN.040	STN.050	STN.060
ROB F LOAD EXOP 30 TO FUTURE - ROB F LOAD EXOP 30 FROM ROB BELTS (10) SPOTS RACK	ROB F REMOVES ASST & POSITIONS TO STATION W/C 1. WELD (12) SPOTS ROB F PLACES ASST ON INTER-OF RACK	ROB H COLLECTS EXOP 30 & POSITIONS TO STATION W/C 1. WELD (10) SPOTS ROB H PLACES PART ON ACCUMULATOR

LEGEND	
	HANDLING
	WELDING
	HANDLING/FED. WELD
	ROBOT WELDING/ HANDLING
	MIG WELDING
	STUD WELD
	ADHESIVE/SEALANT
	TUCKER STUDWELDING
	PROJECTION WELDING
	PEDISTAL WELDER
	DATE STAMP
	ROBOT GEOMETRY WELD FIXTURE
	HOLDING FIXTURE
	DEDICATED GEOMETRY WELD FIXTURE
	ROBOT/DEDICATED GEOMETRY WELD FIXTURE
	MANUAL WELD FIXTURE
	RESPOT FIXTURE
	STILLAGE
	PIPE RAILS
	OPERATOR (+No.)
	MISC. ASSIST

EQUIPMENT REQUIREMENT	
OUTPUT FROM ASSEMBLY LINE	= 300,000 V.P.T
EFFICIENCY/AVAILABILITY	= 85 %
CYCLE TIME	= 36 SECONDS
NO OF SHIFTS	= 2
FLOOR AREA	= 300 SQUARE METRES
NO OF OPERATORS	= 3
NO OF SPOTWELDS	= 54
NO OF PROJECTION WELDS	= 11
NO OF STUDS	= 13
NO OF STUD GUNS	= 2
NO OF GRIPPERS	= 6
NO OF PORTABLE WELD GUNS	= 6
NO OF ROBOTS	= 8
DATE STAMPING OPERATION	= OP. XXX

Lamb Technicon UK

FACILITIES LAYOUT

NOTICE

THE DRAWING IS THE PROPERTY OF
LAMB TECHNICON UK
IT IS TO BE KEPT IN THE
DRAWING OFFICE AND NOT TO BE
REPRODUCED OR COPIED IN ANY
MANNER WITHOUT THE WRITTEN
CONSENT OF LAMB TECHNICON UK

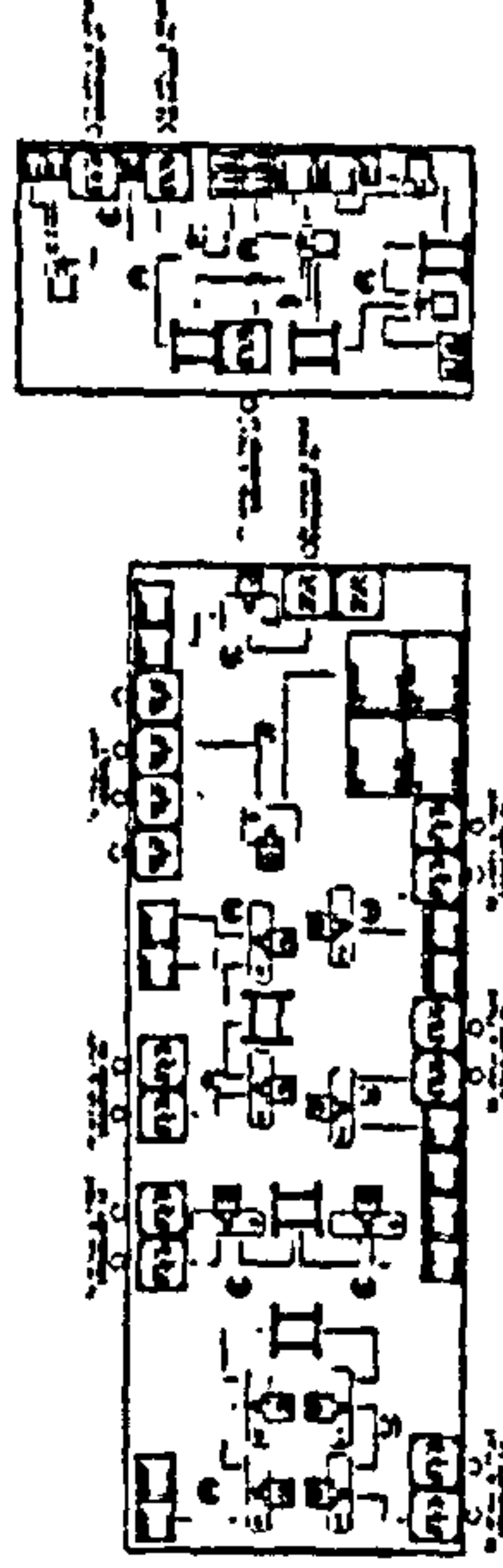
ZONE 4 - PANEL DASH ASSEMBLY
IE No. 9B
DAGENHAM






















PW-1479

REV 1

AO

ZONE 7
PEDESTAL WELD
OPERATIONS



LEGEND	
	HANDLING
	WELDING
	HANDLING/PEP WELD
	ROBOT WELDING/ HANDLING
	MIG WELDING
	STUD WELD
	ADHESIVE/SEALANT
	TUCKER STUDWELDING
	PROJECTION WELDING
	PEDESTAL WELDER
	DATE STAMP
	ROBOT GEOMETRY WELD FIXTURE
	HOLDING FIXTURE
	DEDICATED GEOMETRY WELD FIXTURE
	ROBOT/DEDICATED GEOMETRY WELD FIXTURE
	MANUAL WELD FIXTURE
	RESPOT FIXTURE
	STILLAGE
	PIPE RAILS
	OPERATOR (+No.)
	MECH ASSIST

EQUIPMENT REQUIREMENT	
OUTPUT (MAN ASSEMBLY LINE)	= 300 PPD / PY
EFFICIENCY / AVAILABILITY	= 95 %
CYCLE TIME	= 43 SECONDS
NO OF SAMPLES	= 1
FLOOR AREA	= 8 MAX SQUARE METERS
NO OF OPERATORS	= MAX
NO OF SPOT WELDS	= MAX
NO OF PROJECTION WELDS	= MAX
NO OF STUDS	= MAX
NO OF STUD GUNS	= MAX
NO OF GRAPPERS	= MAX
NO OF PORTABLE WELD GUNS	= MAX
NO OF ROBOTS	= MAX
DATE STAMPING OPERATION	= OF MAX

Lamb Technicon UK

FACILITIES LAYOUT

NOTICE

THIS DRAWING IS THE PROPERTY OF
LAND TELEPHONE CO. U.S.
& CANADA COMPANY
AND SHOULD BE KEPT IN A SAFE PLACE
TO PREVENT LOSS OR THEFT. IF LOST,
IT WILL BE REPAIRED AT THE USER'S
EXPENSE.

IN THE UNITED STATES OF AMERICA
Patented Dec. 16, 1908 by LAND TELEPHONE
CO. U.S. & CANADA COMPANY
NEW YORK, N.Y.

OVERALL LAYOUT

COLORED
PW-1479

AD